HauteWRAP
HR-WRAP Series Plus and WRAPter
Wireless Router
Hardware Manual

Version 4.1

Table of Contents

Contents

Introduction / features 5
Images of HauteWRAP HR-WRAP series 6
The HauteWRAP HR-WRAP Series 7
Antenna Port Utilization in HauteWRAP Family of Products 7
Outdoor System Ports 8
Indoor System Ports 9
Radio Module Nomenclature 11
Product Antenna Mapping Details 12
Basic Installation 15
Warnings 15
Setup Requirements 15
Weatherizing your antenna connections 17
Power Over Ethernet 18
Grounding and Lightening Protection 20
HauteRouterOS Software Basic Setup Guide 21
Basic Configuration Tasks 38
Advanced Configuration Tasks 45
Special Notes on the -9 (902-928MHz) Option 48
Federal Communications Commission Statement 50
CE Declaration of Conformity 51
Technical data 51
HauteRouterOS Basic Setup Guide 63
Logging into the HauteSpot Router 63
Adding Software Packages 64
Navigating The Terminal Console 65
Basic Configuration Tasks 68
Setup Command 69
Basic Examples 71
Advanced Configuration Tasks 74
802.1q Trunk extension over Wireless P2P Link 76
Access and bandwidth limitation 88
Antenna Port Utilization in HauteWRAP Family of Products 91
HauteRouterOS Bandwidth Control 99
General Information 99
Queue Types 108
Interface Default Queues 111
Simple Queues 112
Queue Trees 113
Application Examples 114
HauteRouterOS Basic Setup Guide 125
General Information 125
Logging into the HauteSpot Router 126
Adding Software Packages 127
Navigating The Terminal Console 127
Basic Configuration Tasks 131
Setup Command 132
Basic Examples 133
Advanced Configuration Tasks 137
HauteRouterOS BGP Command Reference 139
HauteRouterOS BGP Routing Filters 143
HauteRouterOS Bridge 145
HauteRouterOS Certificate Management 157
HauteRouterOS Configuration Management 161
Configuring Static IP on a Windows PC 166
Configuring the HauteWRAP for Serial Terminal Server Operation 167
Configuring the HR-WRAPSX as a basic Bridging Access Point v4.0 170
HauteRouterOS DDNS Update Tool 176
Dynamic DNS Update 177
HauteRouterOS DHCP Client and Server 177
HauteRouterOS DNS Client and Cache 196

Dual Setup with OSPF 199
HauteRouterOS EoIP 200
HauteRouterOS Ethernet Interfaces 206
Option Globetrotter HSDPA USB Modem 209
Firmware update using HSconf 215
HauteRouterOS General Interface Settings 219
Healthy System Values for HauteWRAP HR-WRAPSX Systems 221
HauteRouterOS HotSpot Gateway 224
HauteRouterOS HotSpot User AAA 257
HauteRouterOS HSconf 262
HauteRouterOS HTTP Proxy 269
HWMPplus 274
HauteRouterOS ICMP Bandwidth Test 290
HauteRouterOS Interface Bonding 291
HauteRouterOS IP accounting 298
HauteRouterOS IP Pools 301
HauteRouterOS IP Security 303
HauteRouterOS IP Telephony 321
HauteRouterOS IPIP Tunnel Interfaces 355
HauteRouterOS L2TP Interface 358
HauteRouterOS Log Management 368
HauteRouterOS MAC Level Access (Telnet and HSConfig) 372
HauteRouterOS Mangle 374
Maximum Transmission Unit on HauteWRAP Routers 381
MIMO Setup Guide 390
MPLS Overview 398
HauteRouterOS Multicast Routing Protocol 444
HauteRouterOS NAT 451
HauteRouterOS NDP 458
HauteRouterOS Mesh Routing Protocol 463
HauteRouterOS OSPF 479
HauteRouterOS Packet Filtering 495
HTB 517
DSCP based QoS with HTB 522
HauteRouterOS Packet Flow 524
HauteRouterOS Packet Packing 533
HauteRouterOS Packet Sniffer 535
HauteRouterOS Performance Graphing 543
HauteRouterOS PPP and Asynchronous Interfaces 547
HauteRouterOS PPP User AAA 552
HauteRouterOS PPPoE 558
HauteRouterOS PPTP 569
HauteRouterOS RADIUS client 581
Configuring the HauteWRAP family from HauteSpot TDMA-like protocol (TLP) mode to 802.11 591
HauteRouterOS RIP 595
HauteRouterOS Router User AAA 603
HauteRouterOS Routes, Equal Cost Multipath Routing, Policy Routing 608
HauteRouterOS Scripting Host 614
HauteRouterOS Serial Console and Terminal 636
Setting Up HauteRouteOS and FreeRADIUS 640
HauteRouterOS SNMP Service 646
0BTraffic Analysis for HR-WRAPDX(1) 656
HauteRouterOS SOCKS Proxy Server 658
Steps to recover a completely corrupted system: 666
HauteRouterOS System Resource Management 668
HauteRouterOS System Watchdog 677
HauteRouterOS Terminal Console 679
Using HauteSpot TDMA-like protocol (TLP) for IP Video Over Wireless Multipoint to Point Links 687
HauteRouterOS Torch (Realtime Traffic Monitor) 690
HauteRouterOS Traffic Flow 692
Transparently Bridge two Networks using MPLS 697
Transparently Bridge two Networks without using WDS (EoIP) 701
Troubleshooting HR-WRAP 710
HauteRouterOS UPnP 716
HauteRouterOS User Manager 719
User Manager/Customer page 727
User Manager/User page 744
User Manager/User sign up 751

User Manager/User payments 754
Using Syslogd to Diagnose HauteWRAP Issues 774
Using Webcfg for wireless configuration 784
HauteRouterOS Virtual Router Redundancy Protocol 787
HauteRouterOS VLAN 792
HauteRouterOS Web Proxy 796
Wireless Interfaces 805
Creating Wireless VLANS 842
Option Globetrotter HSDPA USB Modem 851
Overview of TDMA Like Protocol v2 protocol 874
Compatibility and coexistence with other wireless protocols 875
TLPv2 vs 802.11 875
TLPv2 vs TLPv1 876
Security in TLPv2 network 879
Power Over Ethernet 879
Security on HauteSpot Networks HauteWRAP Wireless Routers 881
Serial Port Usage 884
Simple Guide to make a HotSpot gateway 895
Transparently Bridge two Networks using MPLS 900
Custom Radio Module Options for HauteSpot WRAP Routers 903

Introduction / features

The HauteWRAP HR-WRAP series of routers are based on a small single board computer optimized for wireless access and network routing applications.

- ✓ Atheros AR71xx or MIPS 32K processor
- ✓ 256 KB cache (64K data + 64K instruction + 128K L2)
- ✓ · 1, 3 or 9 Ethernet channels
- ✓ · 1,2 or 3 wireless radio modules
- ✓ · 32, 64, 128, or 256 MB SDRAM, 64 bit wide for high memory bandwidth
- ✓ · 512 KB flash for HauteBIOS system BIOS on SX and DX models
- ✓ · 11 to 27V DC supply through DC jack or passive power over LAN 1 connector
- ✓ · 1 serial port (DB9 male)
- ✓ · Watchdog timer
- ✓ · Thermal monitor
- ✓ Optional USB 2.0
- ✓ · 3 LEDs.
- ✓ Standard Power Supply is a 90-240VAC to 18VDC 830mA 15Watt supply which can support up to 330 ft cable runs. If you are using a longer cable run, then optional higher voltage power supplies are available.

The HauteWRAP HR-WRAP series of routers all feature either a high strength, weatherized outdoor or a compact, attractive indoor enclosure. Designed to withstand the toughest environments, the outdoor enclosure is a small, light, yet very solid wireless outdoor enclosure. The enclosure is rated NEMA-67 providing complete protection against ingress of dust (6) as well as protection against immersion in water (7).

The HauteWRAP HR-WRAP series includes a 18VDC passive power over Ethernet injector which can accommodate 90-240VAC power input.

Every HauteWRAP HR-WRAP series router comes pre-installed with the HauteRouteOS operating system which supports a broad range of routing and bridging features and functions. The full detail of using the HauteRouterOS system is covered in the HauteRouterOS Users Manual which is available from the HauteSpot Networks Web Site. See http://www.hautespot.net/support for more information. Over 900 pages of user documentation is available. A brief getting started guide is supplied in this manual.

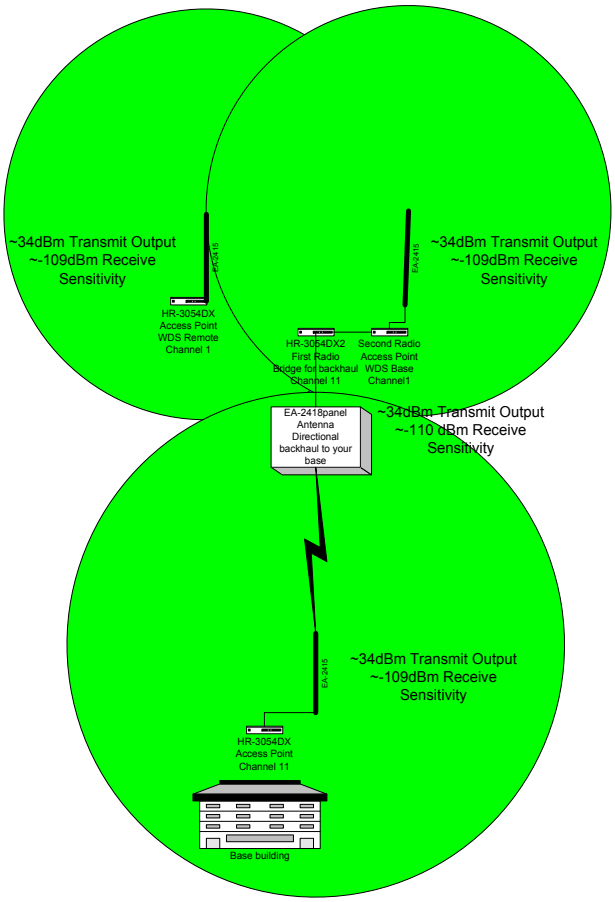
Images of HauteWRAP HR-WRAP series

The HauteWRAP HR-WRAP Series

All of the HauteWRAP HR-WRAP products use the same basic infrastructure, including:

- A NEMA 6 rated, heavy grade aluminum case (8”x6”x3”)
- NP1 type weatherproof RJ-45 connectors for external Ethernet and power connectivity
- Integrated Power Over Ethernet on-board power conversion
- 90-240VAC to 18VDC power transformers standard and Power Over Ethernet Power injector. Options for higher voltage supplies for longer distance runs available.
- Mounting brackets for mounting to 1” – 2” steel pipes or flat surfaces

All of the HauteWRAP HR-WRAP products use N type female bulkhead connectors for antenna connectivity. Pigtail connectors such as the MNLMR400MN-2 cable (not included) are required to attach antennas to the device.



The single radio HauteWRAP HR-WRAPSX and HR-WRAPLX each have one antenna connector at the bottom of the chassis next to the NP1 connector. The single radio units are configured as Client Bridges when they ship from the factory.

Note: A second female N connector is added to the HauteWRAP HR-WRAPSXs, and HR-WRAPLXs serial enabled models with diversity. The second connector allows for a second antenna to be attached to improve receive sensitivity through antenna diversity.

The dual radio HauteWRAP HR-WRAPDX has two antenna connectors, one for each radio.

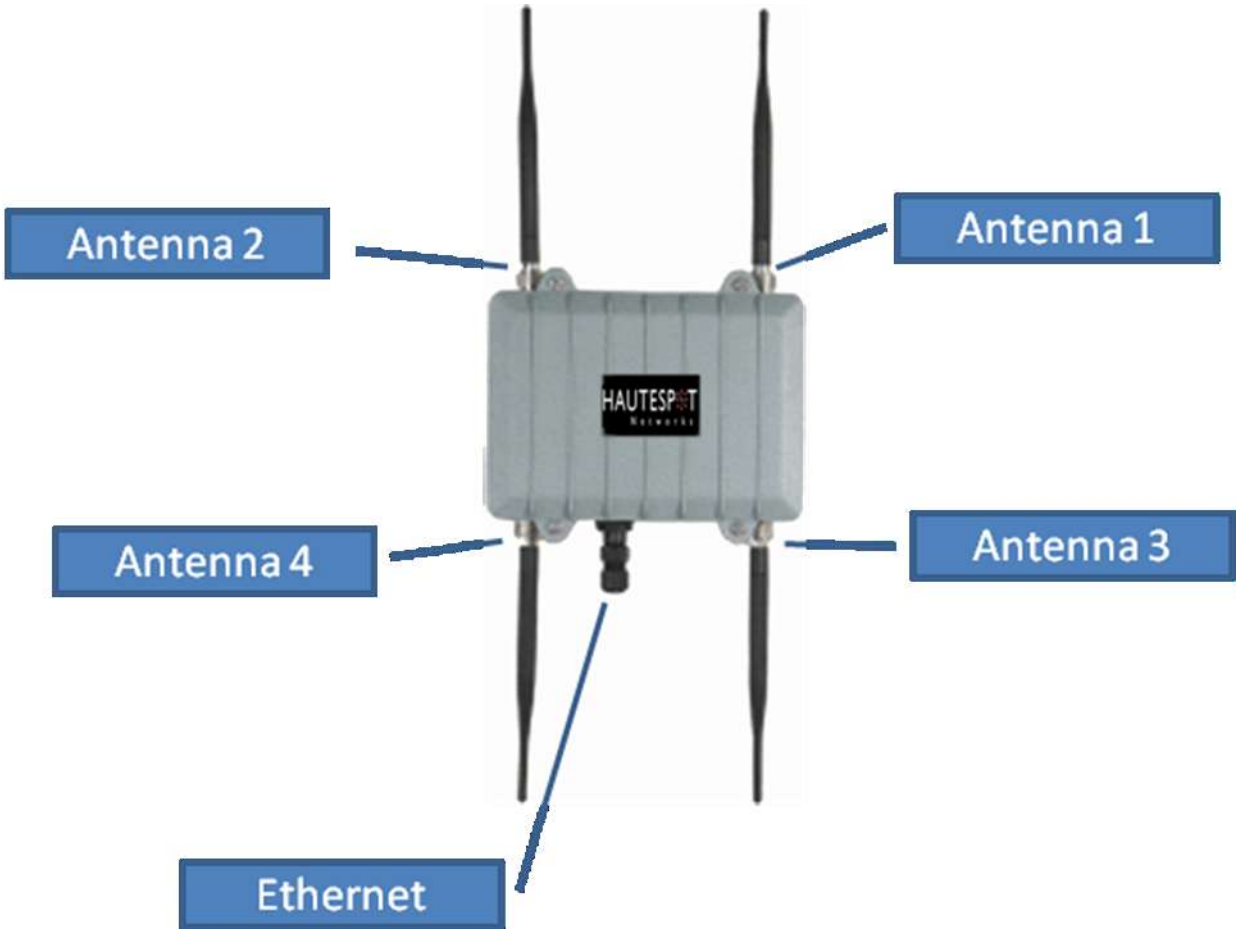
Note: A second connector can be added to each radio interface for a total of 4 connectors as a special request build option at a nominal fee. The second connectors would allow for a second antenna to be attached to improve receive sensitivity through antenna diversity.

Antenna Port Utilization in HauteWRAP Family of Products

There can be anywhere from 1 to 4 antenna ports on any HauteWRAP product. These antenna ports can be assigned to radio antenna ports in a number of ways. This guide provides you with detailed information on antenna port utilization.

Outdoor System Ports

When referring to antenna ports on the outdoor versions of the HauteWRAP we will use the following nomenclature. When looking at an outdoor system from the top down with the front cover facing you and the Ethernet port at the bottom, antenna numbering starts at the upper right corner with antenna 1 and goes right to left, top to bottom, with antenna 2 in the upper left corner, antenna 3 in the lower right corner, and antenna 4 in the lower left corner.



Antenna Port Mapping to Radio Modules – Single Radio Outdoor Systems, No Diversity

When a HauteWRAP outdoor system has a single radio module installed, *without* diversity or MIMO being enabled, the main port, or port A of radio wlan1 as reported by the HauteRouterOS user interface will be connected to antenna 3

Antenna Port Mapping to Radio Modules – Single Radio Outdoor Systems with Diversity or MIMO

When a HauteWRAP outdoor system has a single radio module installed, *with* diversity or MIMO being enabled, the main port, or port A of radio wlan1 as reported by the HauteRouterOS user interface will be connected to antenna 1 and the auxiliary port, or port B of wlan1 as reported by the HauteRouterOS user interface will be connected to antenna 2.

Antenna Port Mapping to Radio Modules – Dual Radio Outdoor Systems, No Diversity

When a HauteWRAP outdoor system has two radio modules installed, *without* diversity or MIMO being enabled, the main port, or port A of radio wlan1 as reported by the HauteRouterOS user interface will be connected to antenna 1 and the main port, or port A of radio wlan2 as reported by the HauteRouterOS user interface will be connected to antenna 2.

Antenna Port Mapping to Radio Modules – Dual Radio Outdoor Systems with Diversity or MIMO

When a HauteWRAP outdoor system has two radio modules installed, with diversity or MIMO being enabled, the main port, or port A of radio wlan1 as reported by the HauteRouterOS user interface will be connected to antenna 1 and the auxiliary port, or port B of wlan1 as reported by the HauteRouterOS user interface will be connected to antenna 2, the main port, or port A of radio wlan2 as reported by the HauteRouterOS user interface will be connected to antenna 4 and the auxiliary port, or port B of wlan2 as reported by the HauteRouterOS user interface will be connected to antenna 3.

Systems with 3G Modems

Antenna Port Mapping to Radio Modules – Single Radio Outdoor Systems, No Diversity, With 3G Modem

When a HauteWRAP outdoor system has a single radio module installed, *without* diversity or MIMO being enabled and includes a 3G modem option, the main port, or port A of radio wlan1 as reported by the HauteRouterOS user interface will be connected to antenna 1 and the 3G modem will be connected to antenna 2.

Antenna Port Mapping to Radio Modules – Single Radio Outdoor Systems with Diversity or MIMO With 3G Modem

When a HauteWRAP outdoor system has a single radio module installed, *with* diversity or MIMO being enabled and includes a 3G modem option, the main port, or port A of radio wlan1 as reported by the HauteRouterOS user interface will be connected to antenna 1 and the auxiliary port, or port B of wlan1 as reported by the HauteRouterOS user interface will be connected to antenna 2, and the 3G modem will be connected to antenna 4. Antenna port 3 is unused.

Antenna Port Mapping to Radio Modules – Dual Radio Outdoor Systems, No Diversity With 3G Modem

When a HauteWRAP outdoor system has two radio modules installed, *without* diversity or MIMO being enabled and includes a 3G modem option, the main port, or port A of radio wlan1 as reported by the HauteRouterOS user interface will be connected to antenna 1 and the main port, or port A of radio wlan2 as reported by the HauteRouterOS user interface will be connected to antenna 2, and the 3G modem will be connected to antenna 4. Antenna port 3 is unused.

Indoor System Ports

For indoor versions of the HauteWRAP, there are up to three antenna ports. Looking at the front panel of the system with the label up and the Ethernet facing you, the left most port is antenna 1, the center port is antenna 2, and the right port is antenna 3

Antenna Port Mapping to Radio Modules – Single Radio Indoor Systems, No Diversity

When a HauteWRAP indoor system has a single radio module installed, *without* diversity or MIMO being enabled, the main port, or port A of radio wlan1 as reported by the HauteRouterOS user interface will be connected to antenna 1.

Antenna Port Mapping to Radio Modules – Single Radio Outdoor Systems with Diversity or MIMO

When a HauteWRAP indoor system has a single radio module installed*, with* diversity or MIMO being enabled, the main port, or port A of radio wlan1 as reported by the HauteRouterOS user interface will be connected to antenna 1 and the auxiliary port, or port B of wlan1 as reported by the HauteRouterOS user interface will be connected to antenna 3.

Antenna Port Mapping to Radio Modules – Dual Radio Outdoor Systems, No Diversity

When a HauteWRAP indoor system has two radio modules installed, *without* diversity or MIMO being enabled, the main port, or port A of radio wlan1 as reported by the HauteRouterOS user interface will be connected to antenna 1 and the main port, or port A of radio wlan2 as reported by the HauteRouterOS user interface will be connected to antenna 3.

Systems with 3G Modems

Antenna Port Mapping to Radio Modules – Single Radio Outdoor Systems, No Diversity, with 3G modem option

When a HauteWRAP indoor system has one radio module installed, without diversity being enabled, the main port, or port A of radio wlan1 as reported by the HauteRouterOS user interface will be connected to antenna 1 and the 3G modem will be connected to antenna 2

Antenna Port Mapping to Radio Modules – Single Radio Outdoor Systems with Diversity or MIMO, With 3G Modem

When a HauteWRAP indoor system has a single radio module installed*, with* diversity or MIMO being enabled, the main port, or port A of radio wlan1 as reported by the HauteRouterOS user interface will be connected to antenna 1 and the auxiliary port, or port B of wlan1 as reported by the HauteRouterOS user interface will be connected to antenna 3, and the 3G modem is connected to antenna 2.

Antenna Port Mapping to Radio Modules – Dual Radio Outdoor Systems, No Diversity, With 3G Modem

When a HauteWRAP indoor system has two radio modules installed, *without* diversity or MIMO being enabled, the main port, or port A of radio wlan1 as reported by the HauteRouterOS user interface will be connected to antenna 1 and the main port, or port A of radio wlan2 as reported by the HauteRouterOS user interface will be connected to antenna 3, and the 3G modem is connected to antenna 2.

Radio Module Nomenclature

Every HauteWRAP product has a set of numbers at the end of it's part code. These numbers indicate the radio modules being using in the router. The first number indicates the radio used for wlan1 and the second number indicates the radio used for wlan2.

-M	2.4 GHz 25dBm/5GHz 24dBm dual band radio
-N	2.4 GHz 25dBm/5GHz 25dBm 2x2 MIMO dual band radio
-2	2.4GHz 28dBm only
-3	3.65GHz 25dBm only
-4	4.9GHz 25dBm only
-5	5GHz 28dBm only
-7	700MHz 28dBm only
-9	900MHz 28dBm only

Product Antenna Mapping Details

The following table shows antenna port mappings for various HauteWRAP products:

Outdoor Products

Product	Antenna Port 1	Antenna Port 2	Antenna Port 3	Antenna Port 4
HR-WRAPSX+M HR-WRAPSX3E+M HR-WRAPLX+M			2.4/5GHz wlan1 A	
HR-WRAPSX+2 HR-WRAPSX3E+2 HR-WRAPLX+2			2.4 GHz wlan1 A	
HR-WRAPSX+37 HR-WRAPSX3E+37 HR-WRAPLX+37			3.65GHz wlan1 A	
HR-WRAPSX+4 HR-WRAPSX3E+4 HR-WRAPLX+4			4.9GHz wlan1 A	
HR-WRAPSX+5 HR-WRAPSX3E+5 HR-WRAPLX+5			5 GHz wlan1 A	
HR-WRAPSX+9 HR-WRAPSX3E+9 HR-WRAPLX+9			900MHz wlan1 A	
HR-WRAPDX+M HR-WRAPDX3E+M	2.4/5GHz wlan1 A	2.4/5GHz wlan2 A		
HR-WRAPDX+2M HR-WRAPDX3E+2M	2.4 GHz wlan1 A	2.4/5GHz wlan2 A		
HR-WRAPDX+37M HR-WRAPDX3E+37M	3.65 GHz wlan1 A	2.4/5GHz wlan2 A		
HR-WRAPDX+4M HR-WRAPDX3E+4M	4.9GHz wlan2 A	2.4/5GHz wlan2 A		
HR-WRAPDX+5M HR-WRAPDX3E+5M	5 GHz wlan2 A	2.4/5GHz wlan2 A		
HR-WRAPDX+9M HR-WRAPDX3E+9M	900MHz wlan2 A	2.4/5GHz wlan2 A		
HR-WRAPDX+22 HR-WRAPDX3E+22	2.4 GHz wlan1 A	2.4 GHz wlan2 A		
HR-WRAPDX+24 HR-WRAPDX3E+24	2.4 GHz wlan1 A	4.9GHz wlan2 A		
HR-WRAPDX+25 HR-WRAPDX3E+25	2.4 GHz wlan1 A	5 GHz wlan2 A		
HR-WRAPDX+29 HR-WRAPDX3E+29	2.4 GHz wlan1 A	900MHz wlan2 A		
HR-WRAPDX+44 HR-WRAPDX3E+44	4.9GHz wlan1 A	4.9GHz wlan2 A		
HR-WRAPDX+45 HR-WRAPDX3E+45	4.9GHz wlan1 A	5 GHz wlan2 A		
HR-WRAPDX+49 HR-WRAPDX3E+49	4.9GHz wlan1 A	900MHz wlan2 A		
HR-WRAPDX+55 HR-WRAPDX3E+55	5 GHz wlan1 A	5 GHz wlan2 A		
HR-WRAPDX+59 HR-WRAPDX3E+59	5 GHz wlan1 A	900MHz wlan2 A		
HR-WRAPDX+99 HR-WRAPDX3E+99	900MHz wlan1 A	900MHz wlan2 A		

Outdoor Products with Diversity or MIMO

Product	Antenna Port 1	Antenna Port 2	Antenna Port 3	Antenna Port 4
HR-WRAPSXs+M	2.4/5GHz wlan1 A	2.4/5GHz wlan1 B		
HR-WRAPSXs+9	900MHz wlan1 A	900MHz wlan1 B		
HR-WRAPSX+N HR-WRAPSX3E+N HR-WRAPLX+N	2.4/5GHz MIMO wlan1 A	2.4/5GHz MIMO wlan1 B		
HR-WRAPDX+N HR-WRAPDX3E+N	2.4/5GHz MIMO wlan1 A	2.4/5GHz MIMO wlan1 B	2.4/5GHz MIMO wlan2 A	2.4/5GHz MIMO wlan2 B
HR-WRAPDX+37N HR-WRAPDX3E+37N	2.4/5GHz MIMO wlan1 A	2.4/5GHz MIMO wlan1 B	3.65GHz wlan2 A	
HR-WRAPDX+4N HR-WRAPDX3E+4N	2.4/5GHz MIMO wlan1 A	2.4/5GHz MIMO wlan1 B	4GHz wlan2 A	
HR-WRAPDX+5N HR-WRAPDX3E+5N	2.4/5GHz MIMO wlan1 A	2.4/5GHz MIMO wlan1 B	5GHz wlan2 A	
HR-WRAPDX+9N HR-WRAPDX3E+9N	2.4/5GHz MIMO wlan1 A	2.4/5GHz MIMO wlan1 B	900MHz wlan2 A	

Outdoor Products with 3G modems

Product	Antenna Port 1	Antenna Port 2	Antenna Port 3	Antenna Port 4
HR-WRAPSX+M3G HR-WRAPSX3E+M3G	2.4/5GHz wlan1 A	3G modem		
HR-WRAPSX+N3G HR-WRAPSX3E+N3G	2.4/5GHz wlan1 A	2.4/5GHz wlan1 B		3G modem
HR-WRAPSX+373G HR-WRAPSX3E+373G	3.65GHz wlan1 A	3G modem		
HR-WRAPSX+93G HR-WRAPSX3E+93G	900MHz wlan1 A	3G modem		
HR-WRAPDX+M3G HR-WRAPDX3E+M3G	2.4/5GHz wlan1 A	2.4/5GHz wlan2 A		3G modem
HR-WRAPDX+37M3G HR-WRAPDX3E+37M3G	3.65 GHz wlan1 A	3.65GHz wlan2 A		3G modem
HR-WRAPDX+4M3G HR-WRAPDX3E+4M3G	4.9GHz wlan2 A	2.4/5GHz wlan2 A		3G modem
HR-WRAPDX+5M3G HR-WRAPDX3E+5M3G	5GHz wlan2 A	2.4/5GHz wlan2 A		3G modem
HR-WRAPDX+9M3G HR-WRAPDX3E+9M3G	900MHz wlan2 A	2.4/5GHz wlan2 A		3G modem

Outdoor Products with MIMO and 3G support

Product	Antenna Port 1	Antenna Port 2	Antenna Port 3	Antenna Port 4
HR-WRAPSX+N3G HR-WRAPSX3E+N3G	2.4/5GHz MIMO wlan1 A	2.4/5GHz MIMO wlan1 B		3G modem
HR-WRAPDX+2N3G HR-WRAPDX3E+2N3G	2.4/5GHz MIMO wlan1 A	2.4/5GHz MIMO wlan1 B	2.4GHz wlan2 A	3G modem
HR-WRAPDX+37N3G HR-WRAPDX3E+37N3G	2.4/5GHz MIMO wlan1 A	2.4/5GHz MIMO wlan1 B	3.65GHz wlan2 A	3G modem
HR-WRAPDX+4N3G HR-WRAPDX3E+4N3G	2.4/5GHz MIMO wlan1 A	2.4/5GHz MIMO wlan1 B	4.9GHz wlan2 A	3G modem
HR-WRAPDX+5N3G HR-WRAPDX3E+5N3G	2.4/5GHz MIMO wlan1 A	2.4/5GHz MIMO wlan1 B	5GHz wlan2 A	3G modem
HR-WRAPDX+9N3G HR-WRAPDX3E+9N3G	2.4/5GHz MIMO wlan1 A	2.4/5GHz MIMO wlan1 B	900MHz wlan2 A	3G modem

Indoor Products

Product	Antenna Port 1	Antenna Port 2	Antenna Port 3
HR-WRAPSXi+M	2.4/5GHz wlan1 A		
HR-WRAPSXi+2	2.4 GHz wlan1 A		
HR-WRAPSXi+37	3.65GHz wlan1 A		
HR-WRAPSXi+4	4.9GHz wlan1 A		
HR-WRAPSXi+5	5 GHz wlan1 A		
HR-WRAPSXi+9	900MHz wlan1 A		
HR-WRAPDXi+M	2.4/5GHz wlan1 A		2.4/5GHz wlan2 A
HR-WRAPDXi+2M	2.4 GHz wlan1 A		2.4/5GHz wlan2 A
HR-WRAPDXi+37M	3.65 GHz wlan1 A		2.4/5GHz wlan2 A
HR-WRAPDXi+4M	4.9GHz wlan2 A		2.4/5GHz wlan2 A
HR-WRAPDXi+5M	5 GHz wlan2 A		2.4/5GHz wlan2 A
HR-WRAPDXi+9M	900MHz wlan2 A		2.4/5GHz wlan2 A
HR-WRAPDXi+22	2.4 GHz wlan1 A		2.4 GHz wlan2 A
HR-WRAPDXi+24	2.4 GHz wlan1 A		4.9GHz wlan2 A
HR-WRAPDXi+25	2.4 GHz wlan1 A		5 GHz wlan2 A
HR-WRAPDXi+29	2.4 GHz wlan1 A		900MHz wlan2 A
HR-WRAPDXi+44	4.9GHz wlan1 A		4.9GHz wlan2 A
HR-WRAPDXi+45	4.9GHz wlan1 A		5 GHz wlan2 A
HR-WRAPDXi+49	4.9GHz wlan1 A		900MHz wlan2 A
HR-WRAPDXi+55	5 GHz wlan1 A		5 GHz wlan2 A
HR-WRAPDXi+59	5 GHz wlan1 A		900MHz wlan2 A
HR-WRAPDXi+99	900MHz wlan1 A		900MHz wlan2 A

Indoor Products with MIMO

Product	Antenna Port 1	Antenna Port 2	Antenna Port 3
HR-WRAPSXi+N	2.4/5GHz MIMO wlan1 A	2.4/5GHz MIMO wlan1 B	
HR-WRAPDX+2N	2.4/5GHz MIMO wlan1 A	2.4/5GHz MIMO wlan1 B	2.4GHz wlan2 A
HR-WRAPDX+37N	2.4/5GHz MIMO wlan1 A	2.4/5GHz MIMO wlan1 B	3.65GHz wlan2 A
HR-WRAPDX+4N	2.4/5GHz MIMO wlan1 A	2.4/5GHz MIMO wlan1 B	4GHz wlan2 A
HR-WRAPDX+5N	2.4/5GHz MIMO wlan1 A	2.4/5GHz MIMO wlan1 B	5GHz wlan2 A
HR-WRAPDX+9N	2.4/5GHz MIMO wlan1 A	2.4/5GHz MIMO wlan1 B	900MHz wlan2 A

Indoor Products with 3G modems

Product	Antenna Port 1	Antenna Port 2	Antenna Port 3
HR-WRAPSXi+M	2.4/5GHz wlan1 A	3G modem	
HR-WRAPSXi+2	2.4 GHz wlan1 A	3G modem	
HR-WRAPSXi+37	3.65GHz wlan1 A	3G modem	
HR-WRAPSXi+4	4.9GHz wlan1 A	3G modem	
HR-WRAPSXi+5	5 GHz wlan1 A	3G modem	
HR-WRAPSXi+9	900MHz wlan1 A	3G modem	
HR-WRAPDXi+M	2.4/5GHz wlan1 A	3G modem	2.4/5GHz wlan2 A
HR-WRAPDXi+2M	2.4 GHz wlan1 A	3G modem	2.4/5GHz wlan2 A
HR-WRAPDXi+37M	3.65 GHz wlan1 A	3G modem	2.4/5GHz wlan2 A
HR-WRAPDXi+4M	4.9GHz wlan2 A	3G modem	2.4/5GHz wlan2 A
HR-WRAPDXi+5M	5 GHz wlan2 A	3G modem	2.4/5GHz wlan2 A
HR-WRAPDXi+9M	900MHz wlan2 A	3G modem	2.4/5GHz wlan2 A
HR-WRAPDXi+22	2.4 GHz wlan1 A	3G modem	2.4 GHz wlan2 A
HR-WRAPDXi+24	2.4 GHz wlan1 A	3G modem	4.9GHz wlan2 A
HR-WRAPDXi+25	2.4 GHz wlan1 A	3G modem	5 GHz wlan2 A
HR-WRAPDXi+29	2.4 GHz wlan1 A	3G modem	900MHz wlan2 A
HR-WRAPDXi+44	4.9GHz wlan1 A	3G modem	4.9GHz wlan2 A
HR-WRAPDXi+45	4.9GHz wlan1 A	3G modem	5 GHz wlan2 A
HR-WRAPDXi+49	4.9GHz wlan1 A	3G modem	900MHz wlan2 A
HR-WRAPDXi+55	5 GHz wlan1 A	3G modem	5 GHz wlan2 A
HR-WRAPDXi+59	5 GHz wlan1 A	3G modem	900MHz wlan2 A
HR-WRAPDXi+99	900MHz wlan1 A	3G modem	900MHz wlan2 A

Indoor Products with MIMO and 3G support

Product	Antenna Port 1	Antenna Port 2	Antenna Port 3
HR-WRAPSXi+N3G	2.4/5GHz MIMO wlan1 A	3G modem	2.4/5GHz MIMO wlan1 B

Basic Installation

Warnings

• Do not work on the system or connect or disconnect cables during periods of lightning activity.

• Do not locate the antenna near overhead power lines or other electric light or power circuits, or where it can come into contact with such circuits. When installing the antenna, take extreme care not to come into contact with such circuits, as they may cause serious injury or death.

• Only trained and qualified personnel should be allowed to install, replace, or service this equipment.

• To meet regulatory restrictions, the radio and the external antenna must be professionally installed. The network administrator or other IT professional responsible for installing and configuring the unit must be a suitable professional installer. Following installation, access to the unit should be password protected by the network administrator to maintain regulatory compliance.

• The HauteWRAP router and PoE injector can be damaged by incorrect power application. Read and carefully follow the installation instructions before connecting the system to its power source.

Setup Requirements

Before starting, please verify that the following is available:

• CAT5/5e or FTP Outdoor Ethernet cable (from the HauteWRAP router to PoE Injector)

• At least one computer is installed with HauteSpot Windows Tools and HSNM along with a wired or wireless network interface adapter. HauteSpot Windows Tools and HSNM can be downloaded from the HauteSpot Networks web site.

• TCP/IP protocol is installed and IP address parameters are properly configured on all your network's nodes

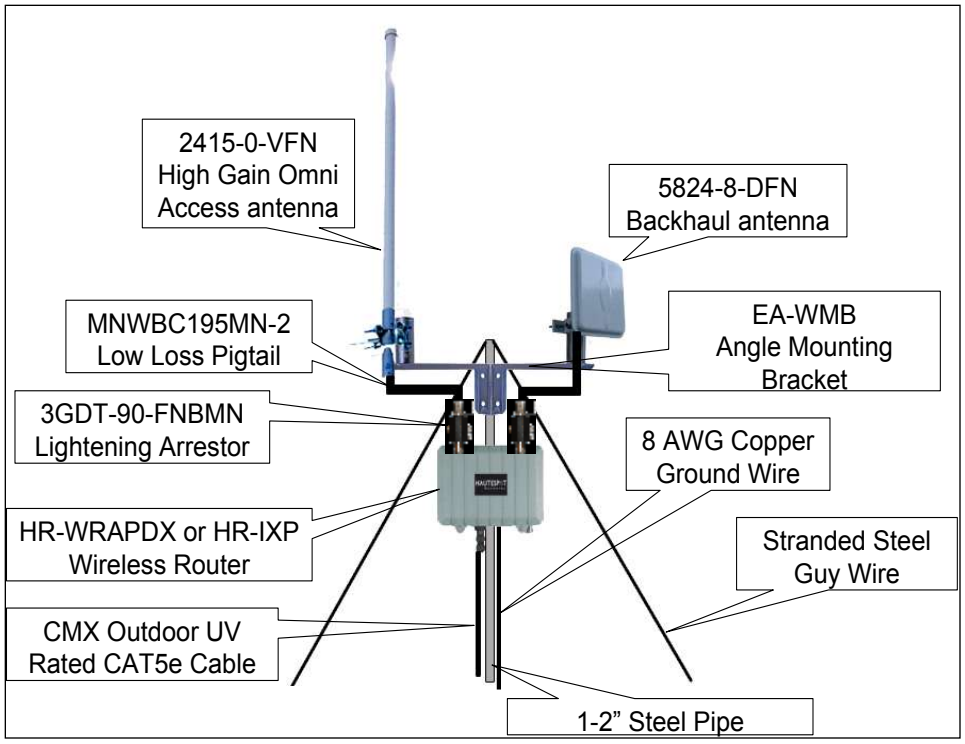
Important!

• Configure and verify the HauteWRAP router operations first before you mount the unit in a remote location.

• You may need to install a lightning arrestor to protect your HauteWRAP router from lightning.

• For choosing the best location for your HauteWRAP router choose an elevated location where trees, buildings and large steel structures will not obstruct the antenna signals and which offers maximum line-of-sight propagation with the users or remote nodes.

• Select an appropriate antenna to improve range and/or coverage. The HauteWRAP router lets you fine-tune parameters such as the transmit power, noise immunity, etc. to achieve the best results. Consideration should be given to Fresnel zone clearance, obstructions, antenna downtilt and other configuration parameters. This manual is not intended to provide instruction on RF system installation design. Please refer to the HauteSpot Networks web site (http://www.hautespot.net/support) for more information and tools to assist you with RF design.



The units are designed to attach to a 1" or 2" steel pole or to a wall. We recommend that a 2" pole be used if possible in order to minimize swaying and other motion. Use the U shaped clamps to attach the unit to the pole with the antenna connectors facing up and Ethernet connector facing down, this will keep water from intruding into the chassis.

When mounting to a pole, first attach the pole bracket to the pole using the included U-bolts, then screw the bracket onto the back of the chassis using the included stainless screws. Make sure that the metal pole is properly grounded and that good electrical contact is made between the bracket and the pole.

Once the unit is mounted to the pole, mount your antenna immediately above your unit. Attach the antenna to the unit using the shortest possible pigtail connector. This will minimize signal loss due to attenuation. If you live in an area where lightening is an issue, we recommend that you use a lightening arrestor as well.

For HauteWRAP HR-WRAPSX standard units there is only one antenna connection. The HauteWRAP HR-WRAPDX unit ships with two antenna connections.

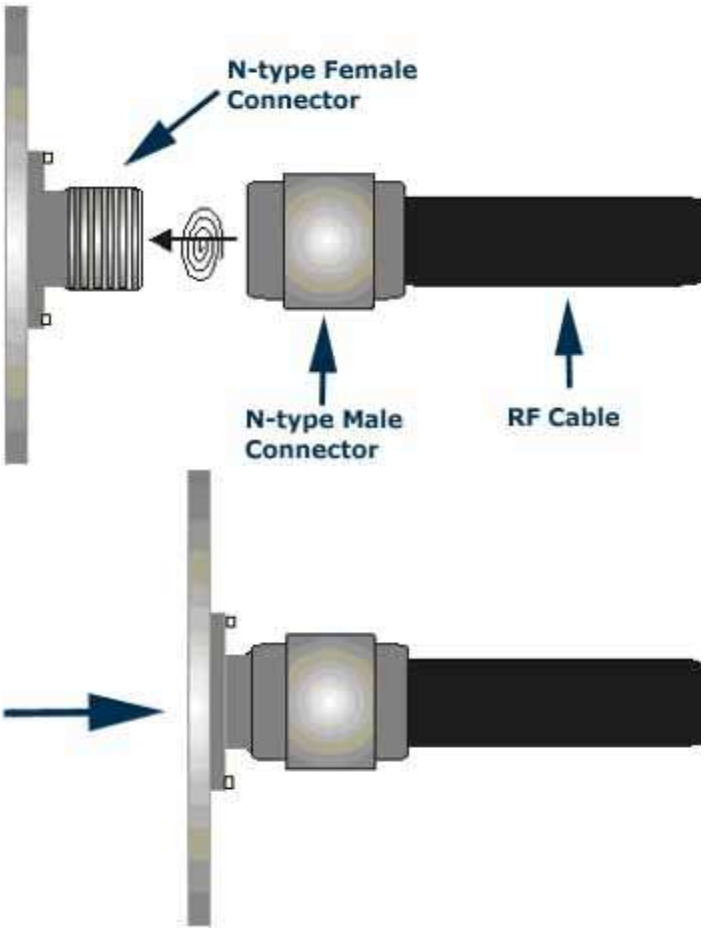
For all units the RJ-45 connector provides both power and Ethernet connectivity for the unit.

The base configuration for these units is for them to ship in access point operation mode.

Weatherizing your antenna connections

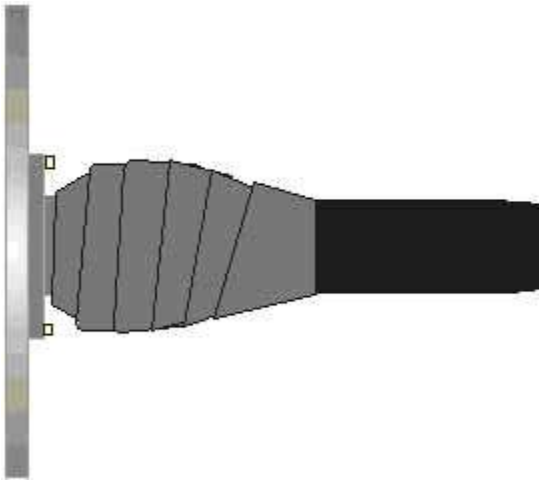
Step 1. Connecting Cable Assembly to Antenna/Enclosure

Before connecting RF cables together, spray the male and female connectors with a electrically lubricating spray such as CRC 2-26 (Order part # 591 from CRC Industries Warminster, PA). This will reduce corrosion and improve electrical contact. Then attach RF cable assembly to antenna/enclosure by fastening the N-type male connector to the N-type female connector. Notes: Ensure both connectors are clean and dry. Hand-tighten firmly.



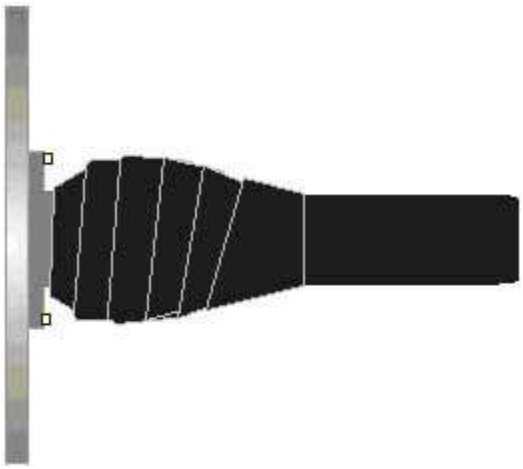
Step 2. Sealing the Connector Assembly

Tightly wrap the connectors with 2 layers of rubber tape (butyl). Notes: Rubber tape should seal entire connection and extend 1 inch beyond antenna connector/cable assembly. CoaX+seal (rubber tape) works very well in this application.



Step 3. Covering Rubber Tape with Electrical Tape

Tightly wrap the rubber tape with 2 layers of electrical tape. Note: Electrical tape should extend 1 inch beyond rubber tape to ensure full coverage.



Power Over Ethernet

WARNING: The HauteWRAP HR-WRAP series of routers are designed specifically for an input voltage of between 11VDC and 27VDC. Using a power supply with an output voltage in excess of 28VDC will damage your system. Such damage is not covered under your product warranty. Only use power supplies provided by HauteSpot Networks.

Power-over-Ethernet (PoE) or "Active Ethernet" eliminates the need to run 90-240 VAC power to Wireless Access Points and other devices on a wired LAN. Using Power-over-Ethernet system installers need to run only a single CAT5 or CAT6 Ethernet cable that carries both power and data to each device. This allows greater flexibility in the locating of AP's and network devices and significantly decreasing installation costs in many cases.

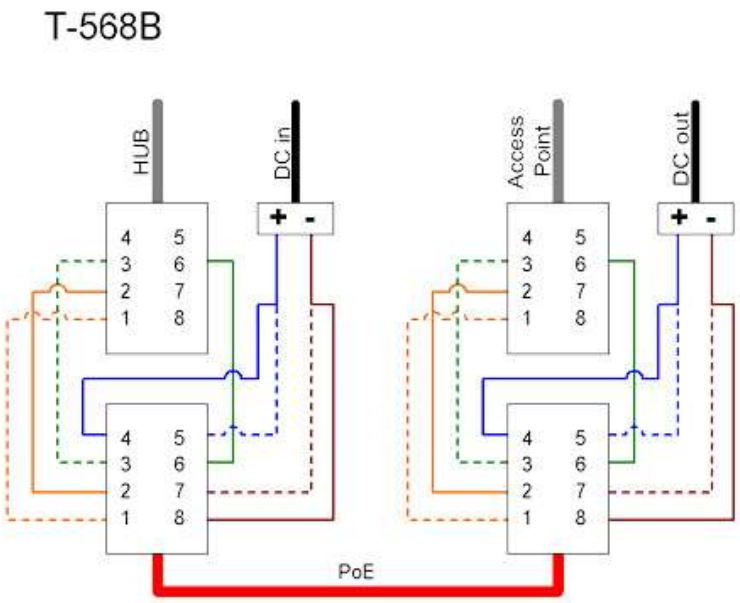
Power-over-Ethernet begins with a CAT5/6 "Injector" that inserts a DC Voltage onto the CAT5/6 cable. The Injector is typically installed in the "wiring closet" near the Ethernet switch or hub.

Our Wireless Access Points accept the injected DC power directly from the CAT5/6 cable through their NP1 (RJ45) jack. Our devices are considered to be "PoE-Compatible" or "Active Ethernet Compatible".

Voltage and Pinout Standards
Although the IEEE has a PoE standard called IEEE 802.3af, different equipment vendors use different PoE voltages and CAT5/6 pin configurations to provide the DC power. Therefore it is important to select the appropriate PoE devices for each piece of equipment you plan to power through the CAT5 cable.

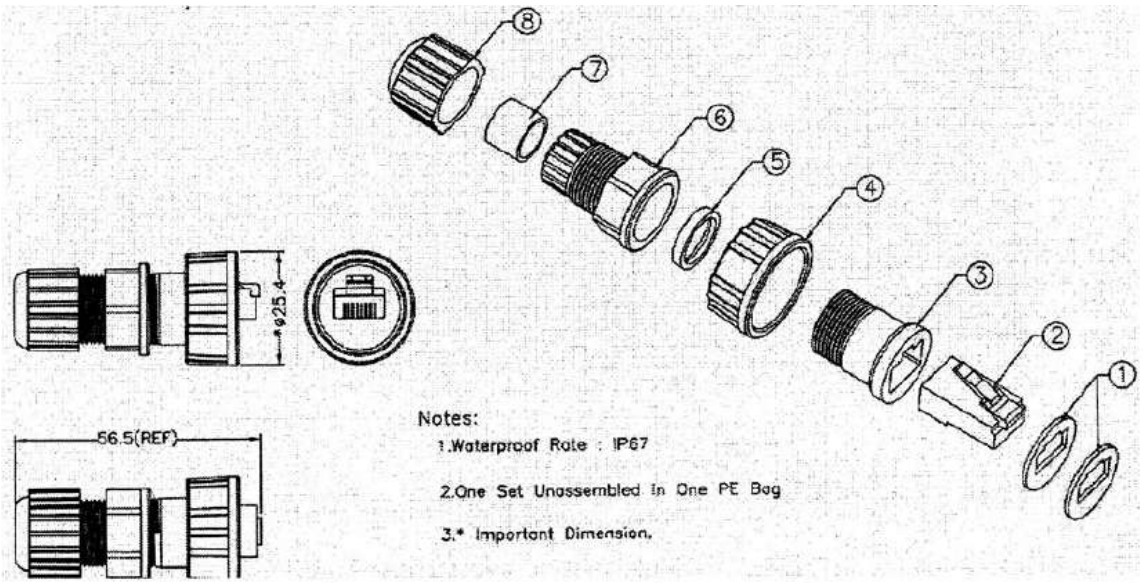
The IEEE has standardized on the use of 48 VDC as the Injected PoE voltage. The HauteWRAP HR-WRAP series cannot accept this higher voltage. Where the maximum cable length has not been a major consideration the included 12 VDC is sufficient (up to 100 feet). For distances over 100 feet use the 18VDC adapter which can be purchased from HauteSpot Networks. **DO NOT USE other PoE solutions in combination with HauteSpot Networks devices.**

All of the HauteWRAP HR-WRAP products follow the IEEE standard for wiring. The following diagram shows how your cable from the injector to the HauteWRAP HR-WRAP unit should be wired. Included in your kit is a male RJ-45 connector and watertight fitting. This is provided so that you can measure the appropriate cable length in order to run up the mounting location of your system. Follow the attached diagram to create your lead wire and then crimp the connector on so that the water tight connector will seal the RJ-45 onto your unit. Power is delivered with positive voltage on pins 4/5 and negative voltage on pins 7/8. This is passive PoE and there is no negotiation. If using an active PoE hub or injector, you must disable active negotiation.



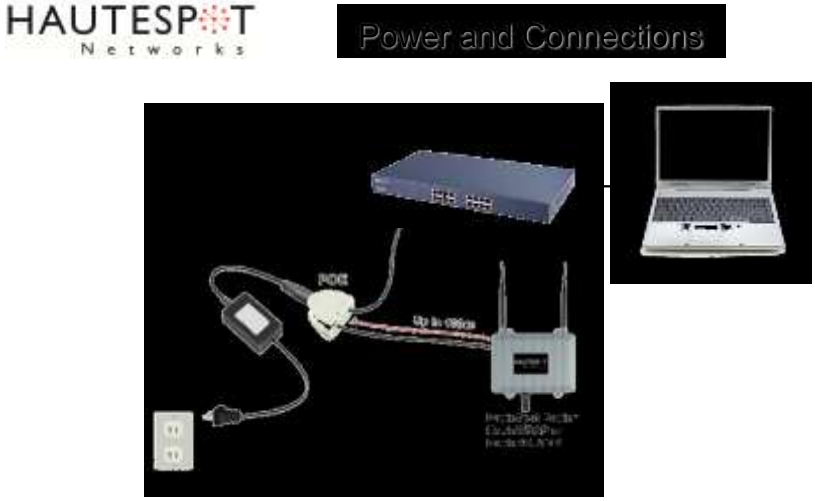
You should use a crimping tool to attach a male RJ-45 connector to your CAT 5/6 cable using the above diagram. This end will go to your POE injector. We recommend the use of 22AWG CAT6 cable if possible, as this will create less wire resistance, increasing the cable distance and improving reliability of your system.

You should use the following diagram to assemble the male end of your NP1 type waterproof connector to the other end of your CAT5/6 cable. This will attach to the HauteWRAP HR-WRAP unit.



Attach your PoE injector as follows:

The Standard PoE transformer will operate on any AC US input voltage from 90-240VAC. It is advisable to use a surge suppressor and/or UPS with this device. Clean power will assure reliable operation.



Copyright © 2008 HauteSpot Networks Corporation

Grounding and Lightening Protection

It is critical that you properly ground your wireless equipment. This means bonding the metal case of your router to a steel pole so that the entire system has a short path to earth ground.

Additional protection may include the use of gas vapor discharge devices, power over Ethernet surge suppression and use of a avalanche clamping UPS.

Complete details on proper grounding may be found in the document “Protecting Your Wireless Investment” which is posted to the HauteSpot Networks web site.

HauteRouterOS Software Basic Setup Guide

HauteSpot RouterOS™ is an independent LinuX+based Operating System for routers and thin routers. It does not require any additional components and has no software pre-requirements. It is designed with a easy-to-use yet powerful interface allowing network administrators to deploy network structures and functions, that would require long education elsewhere simply by following the Reference Manual (and even without it).

HauteSpot RouterOS™ is intended for use on HauteSpot HauteWRAP HR-WRAP family of wireless access routers. It is a full feature routing stack. Remote control with a easy real-time Windows application (HSConf)

- TDMA wireless protocol for video streaming performance
- Simple to configure web interface for installers
- Advance configuration through HSconf client server application
- HSNM, the complete network management system included free.
- Advanced Quality of Service control with burst support
- Stateful firewall with P2P protocol filtering, tunnels and IPsec
- STP bridging with filtering capabilities
- Super high speed 802.11a/b/g wireless with WEP
- WDS and Virtual AP features
- HotSpot for Plug-and-Play access
- HWMP, RIP, OSPF, BGP and OLSR routing protocols
- async PPP with RADUIS AAA

The Guide describes the basic steps of installing and configuring a HauteWRAP router running HauteSpot RouterOS™.

We have created tutorials on-line that walk through the basic interface of the HauteWRAP HR-WRAP and how to configure the most common settings. The tutorials are stored in a protected area of our web site and are only available to customers who have purchased our equipment. Contact HauteSpot Networks to request a user id for this area.

Logging into the HauteSpot HauteWRAP Router using WEB Interface (WEBCFG)

The simplest management interface is to use the Web interface, or Webcfg. This interface is designed for use by virtually any user and should not require extensive training or knowledge. More advanced users will want to use HSconf, since it allows for much more complex configuration.

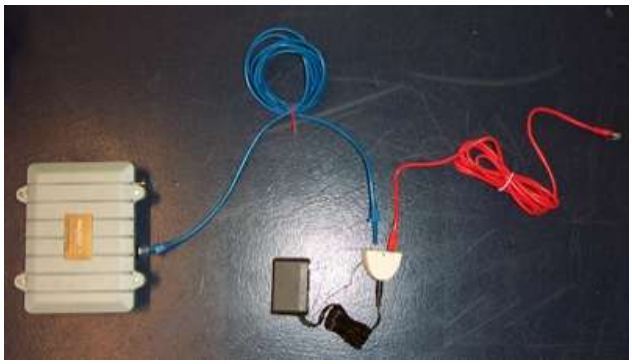
There are two default users that ship from the factory:

"install" is the basic installer user which is limited to only using the web interface.

"admin" is the advanced user who can use both the web interface and HSconf.

The password for either account is blank.

1. The HauteWRAP HR-WRAPSX comes from the factory preconfigured with the Ethernet port assigned to IP address 192.168.1.1. This allows you to connect to the unit using Ethernet in order to configure it for your application requirements.
2. In order to connect to the HauteWRAP HR-WRAPSX, it is best to start with a direct connection between your PC and the HauteWRAP HR-WRAPSX using the included Ethernet cross over cable. Your connection should look like this:

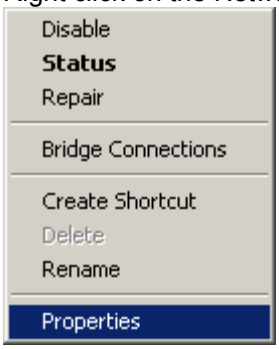


Use a straight through Ethernet cable between the Power+Data port on the PoE injector and the HauteWRAP HR-WRAPSX. Use the red cross over cable between the PoE and your computer.

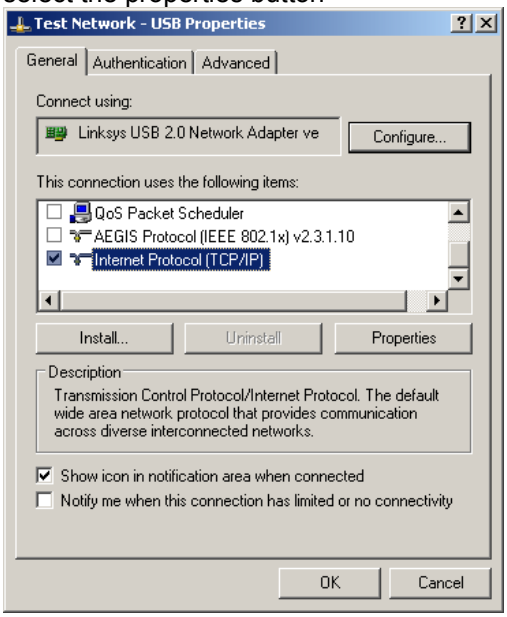
3. Now configure your computer so that it uses IP address 192.168.1.10. Under Windows XP, follow these steps:

a. Go to START | SETTINGS | NETWORK CONNECTIONS and open that window

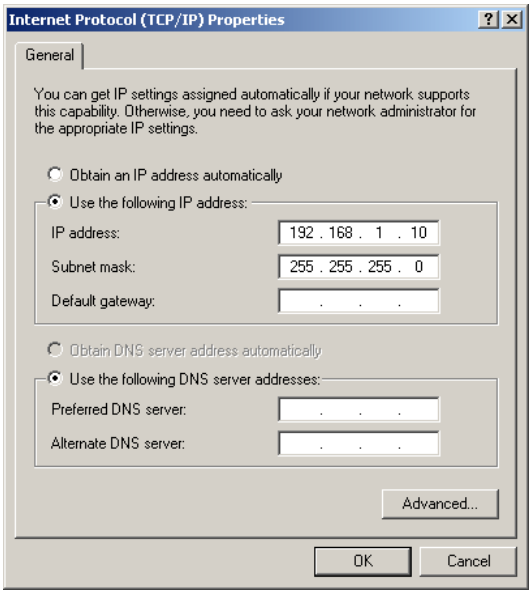
b. Right click on the Network interface for your Ethernet port and select properties



c. Scroll down the window of configuration items and select Internet Protocol. Then select the properties button



d. Set the address to the static address of 192.168.1.10 with a subnet mask of 255.255.255.0 (this is the same as the /24 mask indicator)


Internet Protocol (TCP/IP) Properties
General
You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.
Obtain an IP address automatically
Use the following IP address:
IP address: 192 . 168 . 1 . 10
Subnet mask: 255 . 255 . 255 . 0
Default gateway:
Obtain DNS server address automatically
Use the following DNS server addresses:
Preferred DNS server:
Alternate DNS server:
Advanced...
OK
Cancel

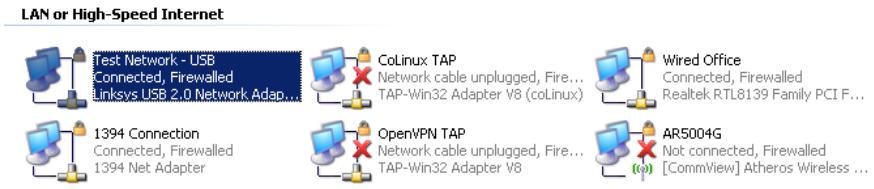
There is no need to set the default gateway or DNS servers at this point.

e. Click OK. You are now configured to attach to the HauteWRAP HR-WRAPSX

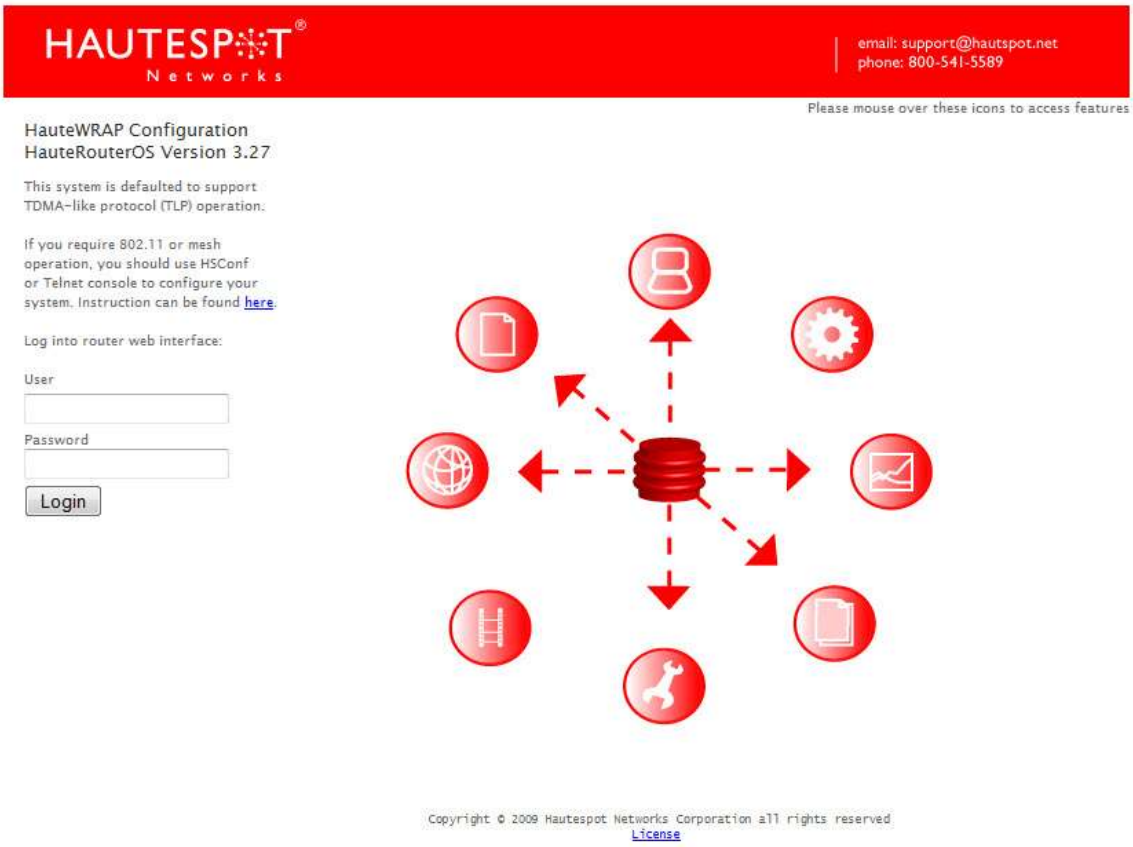
4. Attach the power supply to the PoE injector. There are two indications you should look for to see that the HauteWRAP HR-WRAPSX is working properly:

   a. The green led on the PoE injector is on solid

   b. The link indicator on your PC's Ethernet connector should be on and your computer should report a 100Mbps link.



5. Now open your web browser on your computer to the IP address of the HauteWRAP HR-WRAPSX 192.168.1.1:


HAUTESPOT Networks
email: support@hautspot.net
phone: 800-541-5589
Please mouse over these icons to access features
HauteWRAP Configuration
HauteRouterOS Version 3.27
This system is defaulted to support TDMA-like protocol (TLP) operation.
If you require 802.11 or mesh operation, you should use HSConf or Telnet console to configure your system. Instruction can be found here.
Log into router web interface:
User
Password
Login
Copyright © 2009 Hautespot Networks Corporation all rights reserved
License

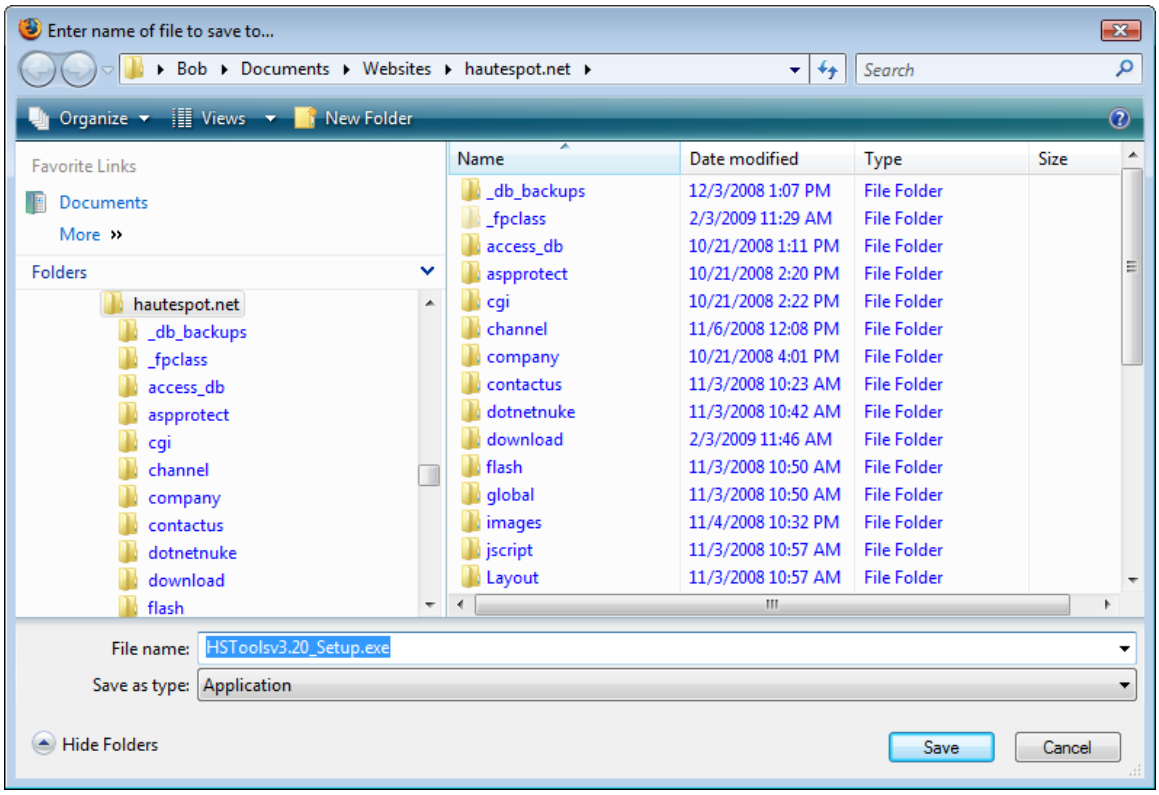
1. Before configuring your system, you should download and install the following 3 items:

Windows Tools – This installation package includes several tools that you will need to configure or repair your system. HS Configurator will be used to configure the HR-WRAP devices in more complex network implementations. Other tools include: HS HotSpot login tool, HS Syslogd, HS Traffic Counter, HS Bandwidth Test, HS Mac Layer Neighbor tool, HS Mac Layer telnet, HS Radio Alignment.

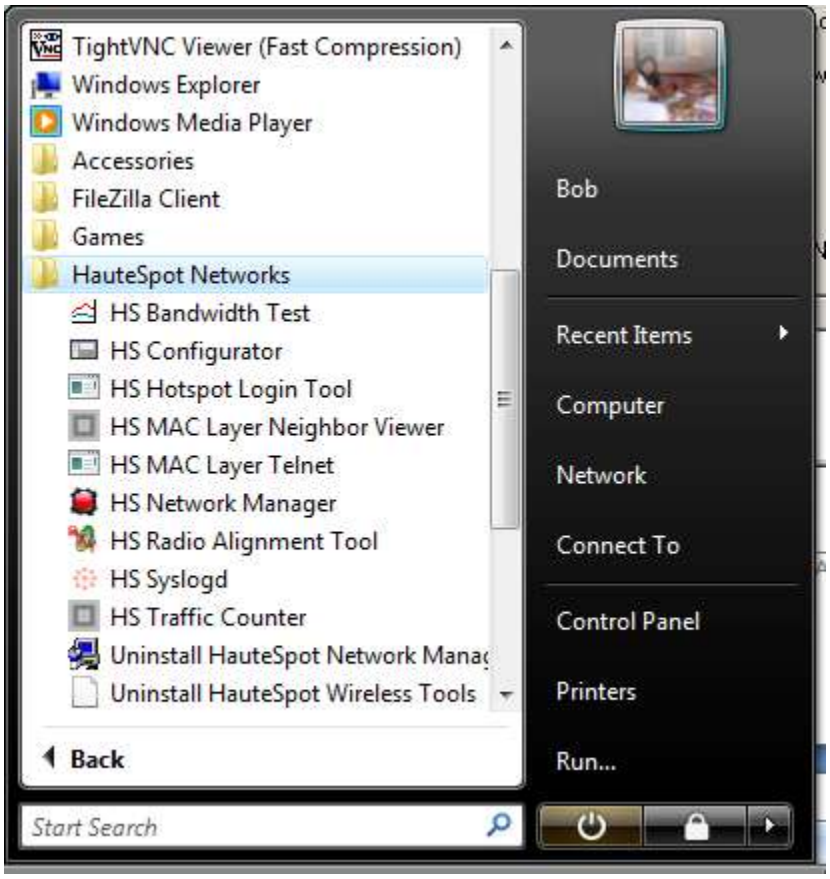
HSNM - MS Windows Network Management System designed specifically for HauteRouterOS. Note that some antivirus tools may report Trojan virus' in HSNM. There is no virus, it is simply the port scanning and discovery features of HSNM. If you have such a conflict, refer to your anti virus software manual for directions on creating an exception to scanning for HSNM.

User Manual – The complete manual with full software guide included.

By clicking on the HSTools download link, you can start the download of the HauteSpot Tools Installer for Windows. Choose **Open** to start the HSTools installer program (you can also save this program to your local disk, and run it from there):



Run the installer. It will add a program group to your MS Windows start menu, which will contain a number of tools, including the HS Configurator (HSconf) tool.



Wireless Interface

1. Once you have completed the download and installation of these utilities, you are ready to view the Web Administration menu (Web Admin). Open a web browser (Mozilla Firefox, Internet Explorer, Google Chrome etc.) and type the default IP address of 192.168.1.1 into the address bar. Login using the username of 'install' with no password.

At the factory your system is set to operate using Time Division Multiple Access (TDMA) protocol by default. This is the best operating protocol for video and voice streaming. While the system will appear to be configured similar to an 802.11 device, it is not using the collision based protocol which 802.11 uses. It will not be able to interoperate with 802.11 devices by default. While the interface will look similar to 802.11, you will need to disable the TDMA protocol if you wish to interoperate with other 802.11 devices. Use HSconf to disable TDMA.

2. **SSID –** This is the system id or group id to which you want the system to associate with. In Station Pseudobridge/wds mode this will be the identifier for the master AP-bridge that you want the unit to associate with. In AP-bridge mode this is the identifier that will be beaconed to all remote slaves.
3. **Mode –** This defines the basic function of your system. The choices are:
    a. **AP-bridge –** This will make the unit a master AP-bridge for a point to multipoint system.
    b. **Station –** This will make the unit a stand-alone slave of the master AP-bridge in a point to multipoint system. In most cases you will **NOT** select this mode.
    c. **Station-wds –** This will make your system a remote slave station using the WDS operating mode for point to multipoint operation. This is the most common client configuration mode.
    d. **Bridge –** This will make your system a peer for a point to point system. Generally you will **NOT** select this mode.
    e. **Station-pseudobridge –** This will make your system a remote slave to the master AP-bridge in a point to multipoint system.

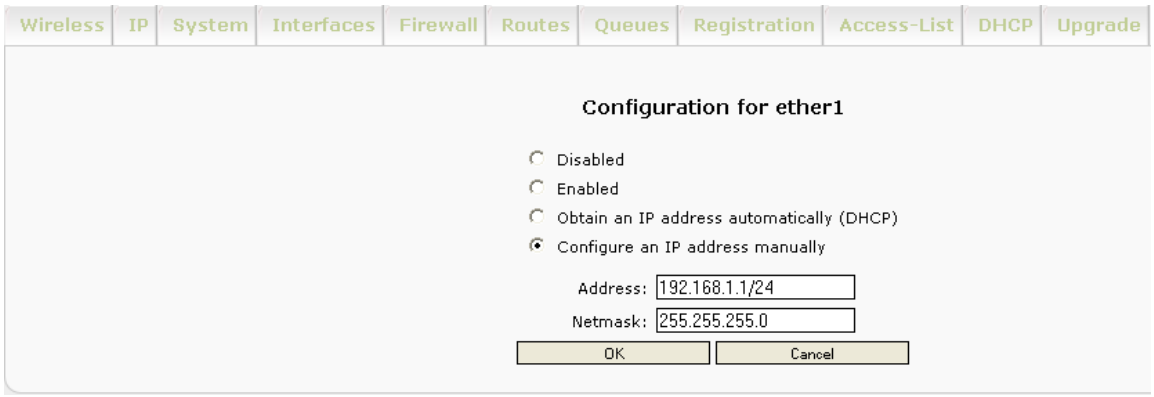
    Version 3.23 and prior had a default mode of Sation Psuedobridge. Post 3.23 devices are defaulted to use Station-wds mode.. To retain Station-Pseudobridge/wds mode you must select it from the drop down menu **EVERY TIME** when making any changes on this page. Once the system switches to Station or AP-Bridge mode it must be reset in order to return to Station Pseudobridge/wds mode.
4. **Band –** In a HauteWRAP that supports multiple bands, this is where you can specify which frequency band to operate. Note that “turbo” means to use a 40MHz channel width.
5. **Frequency –** When the system is operating in AP-Bridge or Bridge mode, you will also see a field for Frequency. This is the center frequency that you want the system to operate. Slave systems will select this frequency automatically based on SSID.

6. **WPA –** Wireless Protected Access security can be enabled here. This security uses a preshared key which is common to all systems in the group. If other types of security are required, you must use HSCONFIG to configure.

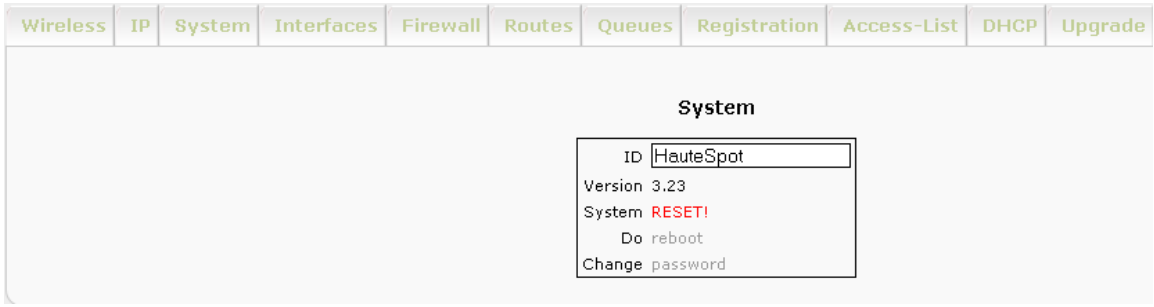
Once all values are set, clicking OK will apply the changes. Remember that if you are configuring for remote slave, you MUST CHANGE THE DROP DOWN MENU ITEM FOR MODE TO STATION-PSEUDOBRIDGE TO RETAIN THIS OPERATING MODE. Once set to any other value you must reset the system to re-enable Station-Pseudobridge mode.

IP

The IP menu allows you to set the IP address of the system and subnet mask. The IP address can be set statically, assigned through DHCP, or not assigned (NOT RECOMMENDED). The IP address should be set with the Netmask following the address (/24 for class C networks).



System



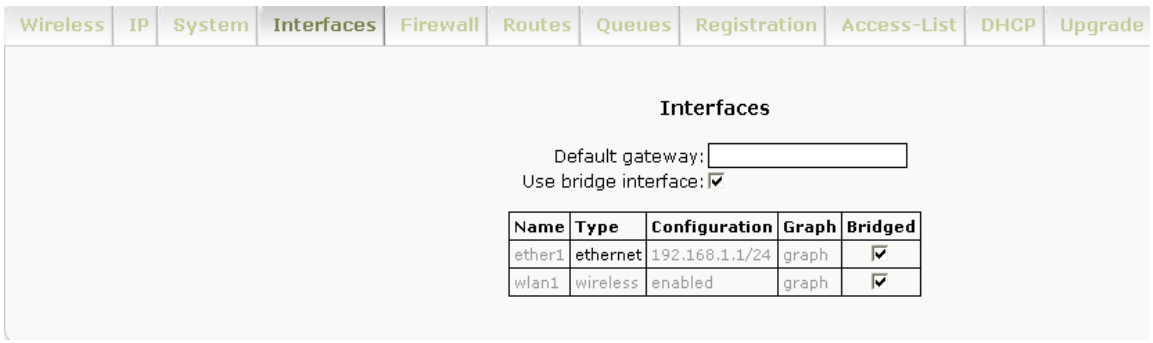
Password changes are for the current user, so if you are logged in as “install” you will be changing the password for this user only.

Rebooting the system will keep the current settings and reboot the system

Resetting the system will revert the system to factory defaults. Since the system may have been installed on a remote tower or pole, the system will give you the option to set an IP address to the unit so that it will remain remotely accessible.

The IP address is applied to the ether1 address. The bridge1 interface will remain with the default IP address of 192.168.1.1. Once the system is reset, you should log in using HSCONFIG and reset the IP addresses as necessary to meet your configuration requirements.

Interfaces



If your system is on the public internet or other routed network, you can set the gateway router address here. From here you can also remove interfaces from the bridge or disable the bridge entirely. Making changes to the bridge could make the device unaccessible.

Firewall

Firewall configuration is one of the more complex Webcfg items. In order to use the Firewall feature the bridge must be disabled with both the wlan1 and ether1 interfaces statically addressed. Firewall is used for NAT (network address translation) and is very common in large public networks.

Public Interface: The public interface is the interface that is connected to the 'Internet'. Your public interface may be either wlan1 or ether1. The other interface will be your masqueraded private network.

The **Protect LAN** and **Protect router** options are simply additional firewall rules in order to block inbound data. **Protect LAN** will block all inbound sessions from the public interface. This will protect your private network from malicious users operating on the public network. **Protect router** is used to block all inbound sessions trying to access the HR-WRAP router itself.

Queues

Queues can be used as a simple QOS (quality of service) management rules. Queues are rarely used by anyone since the HR-WRAP device has other QOS technologies enabled by default.

Registration

The registration window will show you the mac address of the registered clients/ap-brides as well as the associated data rate. The data rate can be helpful to indicate the quality of the link. From the registration window you can also copy entries into the Access-list to be whitelisted or blacklisted.

Access-List

Mac-Address: This is the mac address of the device being blacklisted or whitelisted. If being copied from the Registration window the mac address field will be automatically filled out.

Interface: If operating on a dual radio unit the interface can be switched from wlan1 to wlan2. This dictates which interface blacklistes/whitelists the registered device. Wlan1 will be used the most often.

Authenticate: This selection is the blacklist/whitelist option. To blacklist select 'No' in the drop down list. To whitelist the device select the 'Yes' drop down option.

Forward: The Forward option dictates whether the associated client can access other assosiated clients. If forward is set to 'No' then the client can not initiate any outgoing sessions to other devices. Note that if an access -list entry has an Authenticate mode of 'No' then the Forward option will be irrelevant.

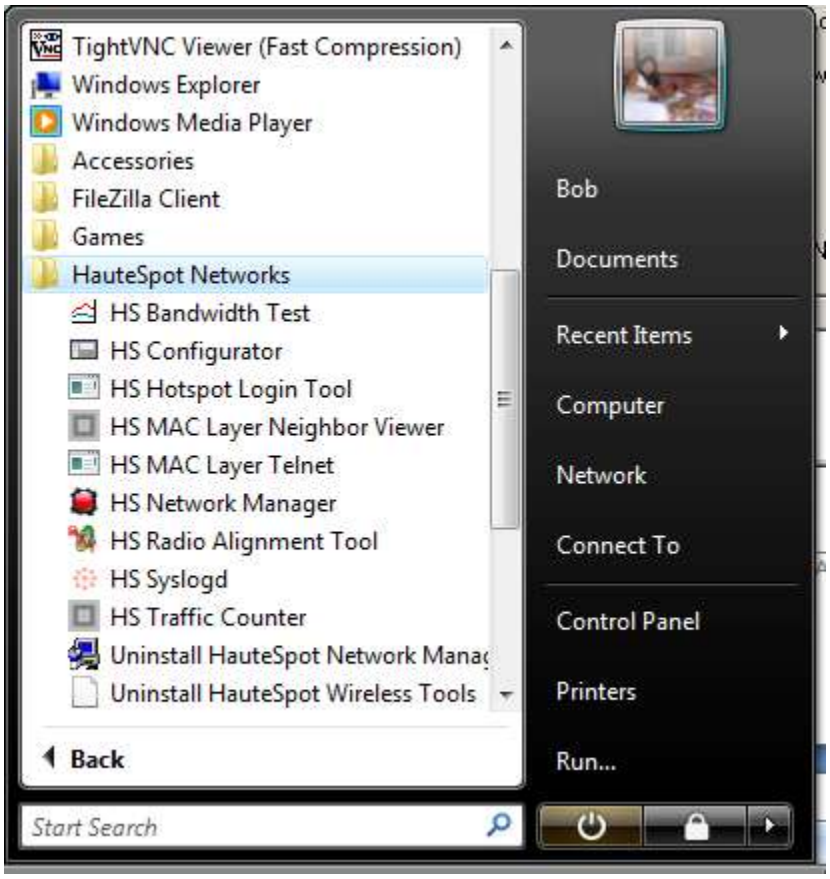
DHCP

The HR-WRAP comes by default with a DHCP server ready to be enabled. DHCP configuration is very straight forward and is commonly used in WISP networks. DHCP is used to automatically assign IP address's to clients instead of dealing with several statically assigned IP's. When using DHCP the user must specify an IP address range and a DHCP gateway. The IP address range is defaulted to be from 192.168.1.2-192.168.1.254. This would allow 253 DCHP users to associate.

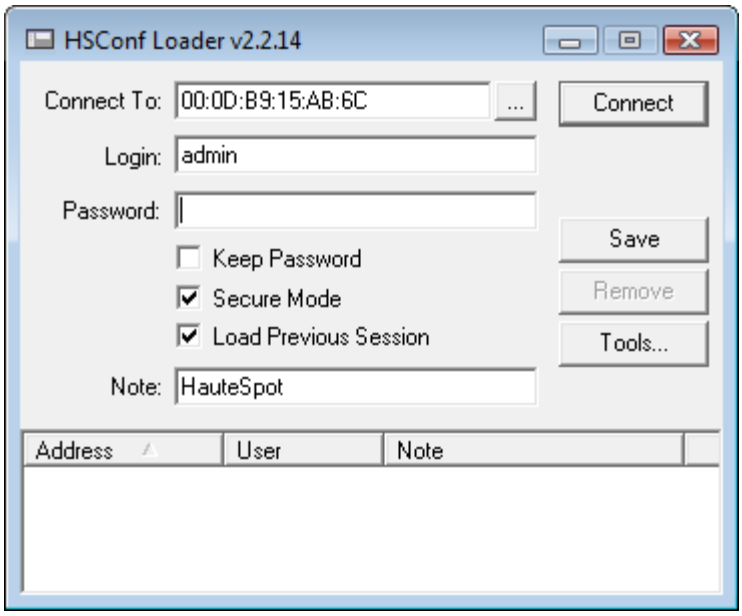
Logging into the HauteSpot HauteWRAP Router using HSConf client server application interface

Most users will want to use the more advanced HSconf, since it allows for much more complex configuration..

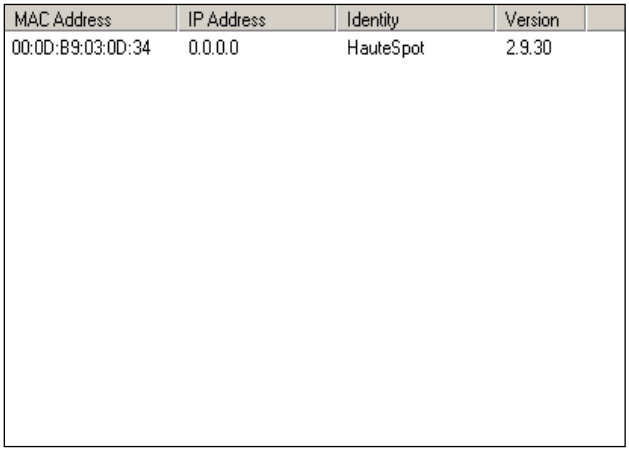
You will have run the HSTools installer from your router's web interface. It will add a program group to your MS Windows start menu, which will contain a number of tools, including the HSconf tool.



The HS Configurator menu item will launch the **HSconf.exe** program and opens the HSconf login window.

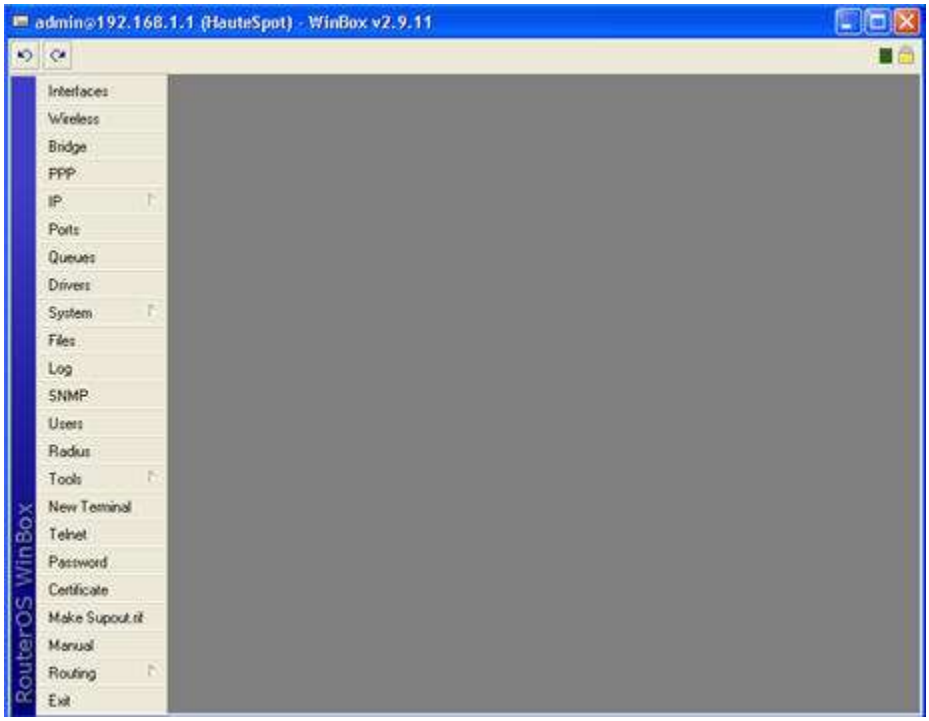


You can search for the router you wish to configure by clicking on the ... button which is located on the right of the Connect To: box. This will open a list of available routers found through layer 2 (MAC) detection. This is the most reliable way to connect to a router for configuration. Left click on the router you wish to manage, set the Login name to “admin” and leave the Password blank, and then click the connect button.



You can also save some sessions to the list (to run them, just double-click on an item).

The HSconf Console of the router:



The HSconf Console uses TCP port 3986 (not secure) or 3987 (secure; requires **security** package to be installed). After logging onto the router you can work with the HauteRoute router's configuration through the HSconf console and perform the same tasks as using the regular console.

Overview of Common Functions

You can use the menu bar to navigate through the router's configuration menus, open configuration windows. By double clicking on some list items in the windows you can open configuration windows for the specific items, and so on.

There are some hints for using the HSconf Console:

- To open the required window, simply click/double click on the corresponding menu item
-

  Add a new entry
-

  Remove an existing entry
-

  Enable an item
-

Disable an item

-

  Make or edit a comment

-

  Refresh a window

-

  Undo an action

-

  Redo an action

-

  Logout from the HSconf Console

Troubleshooting

I cannot open the HSconf Console

- Open a terminal session to the router. Then check the port and address for **www** service in **/ip service print** list. Make sure the address you are connecting from matches the network you've specified in **address** field and that you've specified the correct port in the HSconf loader.
- The command **/ip service set www port=80 address=0.0.0.0/0** will change these values to the default ones so you will be able to connect specifying just the correct address of the router in the address field of HSconf loader
- The HSconf Console uses TCP port 3986 (not secure) or 3987 (secure; requires **security** package to be installed). Make sure you have access to it through the firewall.

Using Telnet

When logging into the router via terminal console, you will be presented with the HauteRoute RouterOS™ login prompt. Use 'admin' and no password (hit 'Enter') for logging in the router for the first time, for example:

```
HauteRoute v4.5
Login: admin
Password:
```

The password can be changed with the **/password** command.

```
[admin@HauteRoute] > password
old password:
new password: ************
```

```
retype new password: ************
[admin@HauteRoute] >
```

Navigating The Terminal Console

Welcome Screen and Command Prompt

After logging into the router you will be presented with the HauteRoute RouterOS™ Welcome Screen and command prompt, for example:

```
  HauteRoute RouterOS 4.5 (c) 1999-2010       http://www.hautespot.net/

Terminal xterm detected, using multiline input mode
[admin@HauteRoute] >
```

The command prompt shows the identity name of the router and the current menu level, for example:

```
[admin@HauteRoute] >                                  Base menu level
[admin@HauteRoute] interface>                         Interface
management
[admin@HauteRoute] ip address>                        IP address
manangement
```

Commands

The list of available commands at any menu level can be obtained by entering the question mark '?', for example:

```
[admin@HauteRoute] >

    certificate  Certificate management
         driver  Driver manageent
           file  Local router file storage.
         import  Run exported configuration script
      interface  Interface configuration
            log  System logs
       password  Change password
           ping  Send ICMP Echo packets
           port  Serial ports
           quit  Quit console
         radius  Radius client settings
           redo  Redo previosly undone action
          setup  Do basic setup of system
           snmp  SNMP settings
  special-login  Special login users
           undo  Undo previous action
           user  User management
             ip  IP options
          queue  Bandwidth management
         system  System information and utilities
           tool  Diagnostics tools
         export  Print or save an export script that can be used to
restore
                 configuration
```

```
[admin@HauteRoute] >
[admin@HauteRoute] ip>

      accounting  Traffic accounting
         address  Address management
             arp  ARP entries management
             dns  DNS settings
        firewall  Firewall management
        neighbor  Neighbors
         packing  Packet packing settings
            pool  IP address pools
           route  Route management
         service  IP services
  policy-routing  Policy routing
            upnp  Universal Plug and Play
            vrrp  Virtual Router Redundancy Protocol
           socks  SOCKS version 4 proxy
         hotspot  HotSpot management
           ipsec  IP security
       web-proxy  HTTP proxy
          export  Print or save an export script that can be used to
restore
                  configuration
[admin@HauteRoute] ip>
```

The list of available commands and menus has short descriptions next to the items. You can move to the desired menu level by typing its name and hitting the [Enter] key, for example:

```
[admin@HauteRoute] >                      Base level menu
[admin@HauteRoute] > driver               Enter 'driver' to move to the
driver level
                                        menu
[admin@HauteRoute] driver> /              Enter '/' to move to the base
level menu
                                        from any level
[admin@HauteRoute] > interface            Enter 'interface' to move to
the interface
                                        level menu
[admin@HauteRoute] interface> /ip         Enter '/ip' to move to the IP
level menu
                                        from any level
[admin@HauteRoute] ip>
```

A command or an argument does not need to be completed, if it is not ambiguous. For example, instead of typing **interface** you can type just **in** or **int**. To complete a command use the **[Tab]** key.

The commands may be invoked from the menu level, where they are located, by typing its name. If the command is in a different menu level than the current one, then the command should be invoked using its full (absolute) or relative path, for example:

```
[admin@HauteRoute] ip route> print                  Prints the routing
table
[admin@HauteRoute] ip route> .. address print       Prints the IP
address table
```

```
[admin@HauteRoute] ip route> /ip address print      Prints the IP
address table
```

The commands may have arguments. The arguments have their names and values. Some commands, may have a required argument that has no name.

Summary on executing the commands and navigating the menus

Command	Action
command [Enter]	Executes the command
[?]	Shows the list of all available commands
command [?]	Displays help on the command and the list of arguments
command argument [?]	Displays help on the command's argument
[Tab]	Completes the command/word. If the input is ambiguous, a second [Tab] gives possible options
/	Moves up to the base level
/command	Executes the base level command
..	Moves up one level
""	Specifies an empty string
"word1 word2"	Specifies a string of 2 words that contain a space

You can abbreviate names of levels, commands and arguments.

For the IP address configuration, instead of using the 'address' and 'netmask' arguments, in most cases you can specify the address together with the number of true bits in the network mask, i.e., there is no need to specify the 'netmask' separately. Thus, the following two entries would be equivalent:

```
/ip address add address 10.0.0.1/24 interface ether1
/ip address add address 10.0.0.1 netmask 255.255.255.0 interface ether1
```

Notes

You must specify the size of the network mask in the address argument, even if it is the 32-bit subnet, i.e., use **10.0.0.1/32** for `address=10.0.0.1 netmask=255.255.255.255`

Basic Configuration Tasks

Interface Management

Before configuring the IP addresses and routes please check the **/interface** menu to see the list of available interfaces. If you have Plug-and-Play cards installed in the router, it is most likely that the device drivers have been loaded for them automatically, and the relevant interfaces appear on the **/interface print** list, for example:

```
[admin@HauteRoute] interface> print
```

```
Flags: X - disabled, D - dynamic, R - running
 #    NAME                          TYPE             RX+RATE    TX+RATE
MTU
 0  R ether1                        ether            0          0
1500
 1  R ether2                        ether            0          0
1500
 2 X  wavelan1                      wavelan          0          0
1500
 3 X  prism1                        wlan             0          0
1500
[admin@HauteRoute] interface>
```

The interfaces need to be enabled, if you want to use them for communications. Use the **/interface enable name** command to enable the interface with a given name or number, for example:

```
[admin@HauteRoute] interface> print
Flags: X - disabled, D - dynamic, R - running
 #    NAME                          TYPE             RX+RATE    TX+RATE
MTU
 0 X  ether1                        ether            0          0
1500
 1 X  ether2                        ether            0          0
1500
[admin@HauteRoute] interface> enable 0
[admin@HauteRoute] interface> enable ether2
[admin@HauteRoute] interface> print
Flags: X - disabled, D - dynamic, R - running
 #    NAME                          TYPE             RX+RATE    TX+RATE
MTU
 0  R ether1                        ether            0          0
1500
 1  R ether2                        ether            0          0
1500
[admin@HauteRoute] interface>
```

The interface name can be changed to a more descriptive one by using **/interface set** command:

```
[admin@HauteRoute] interface> set 0 name=Local; set 1 name=Public
[admin@HauteRoute] interface> print
Flags: X - disabled, D - dynamic, R - running
 #    NAME                          TYPE             RX+RATE    TX+RATE
MTU
 0  R Local                         ether            0          0
1500
 1  R Public                        ether            0          0
1500
[admin@HauteRoute] interface>
```

Notes

The device drivers for NE2000 compatible ISA cards need to be loaded using the **add** command under the /drivers menu. For example, to load the driver for a card with IO address 0x280 and IRQ 5, it is enough to issue the command:

```
[admin@HauteRoute] driver> add name=ne2k-isa io=0x280
[admin@HauteRoute] driver> print
Flags: I - invalid, D - dynamic
 #   DRIVER                                  IRQ IO       MEMORY   ISDN-
PROTOCOL
  0 D RealTek 8139
  1 D Intel EtherExpressPro
  2 D PCI NE2000
  3   ISA NE2000                            280
  4   Moxa C101 Synchronous                           C8000
[admin@HauteRoute] driver>
```

There are some other drivers that should be added manually. Please refer to the respective manual sections for the detailed information on how drivers are to be loaded.

Setup Command

Command name: ***/setup***

The initial setup of the router can be done by using the **/setup** command which offers the following configuration:

- reset all router configuration
- load interface driver
- configure ip address and gateway
- setup dhcp client
- setup dhcp server
- setup pppoe client
- setup pptp client

Configure IP address on router, using the Setup command

Execute the **/setup** command from command line:

```
[admin@HauteRoute] > setup
  Setup uses Safe Mode. It means that all changes that are made during
setup
are reverted in case of error, or if Ctrl-C is used to abort setup. To
keep
changes exit setup using the 'x' key.

[Safe Mode taken]
  Choose options by pressing one of the letters in the left column,
before
dash. Pressing 'x' will exit current menu, pressing Enter key will
select the
entry that is marked by an '*'. You can abort setup at any time by
pressing
Ctrl-C.
Entries marked by '+' are already configured.
Entries marked by '-' cannot be used yet.
Entries marked by 'X' cannot be used without installing additional
packages.
   r - reset all router configuration
```

```
 + l - load interface driver
 * a - configure ip address and gateway
   d - setup dhcp client
   s - setup dhcp server
   p - setup pppoe client
   t - setup pptp client
   x - exit menu
your choice [press Enter to configure ip address and gateway]: a
```

To configure IP address and gateway, press **a** or [Enter], if the **a** choice is marked with an asterisk symbol ('*').

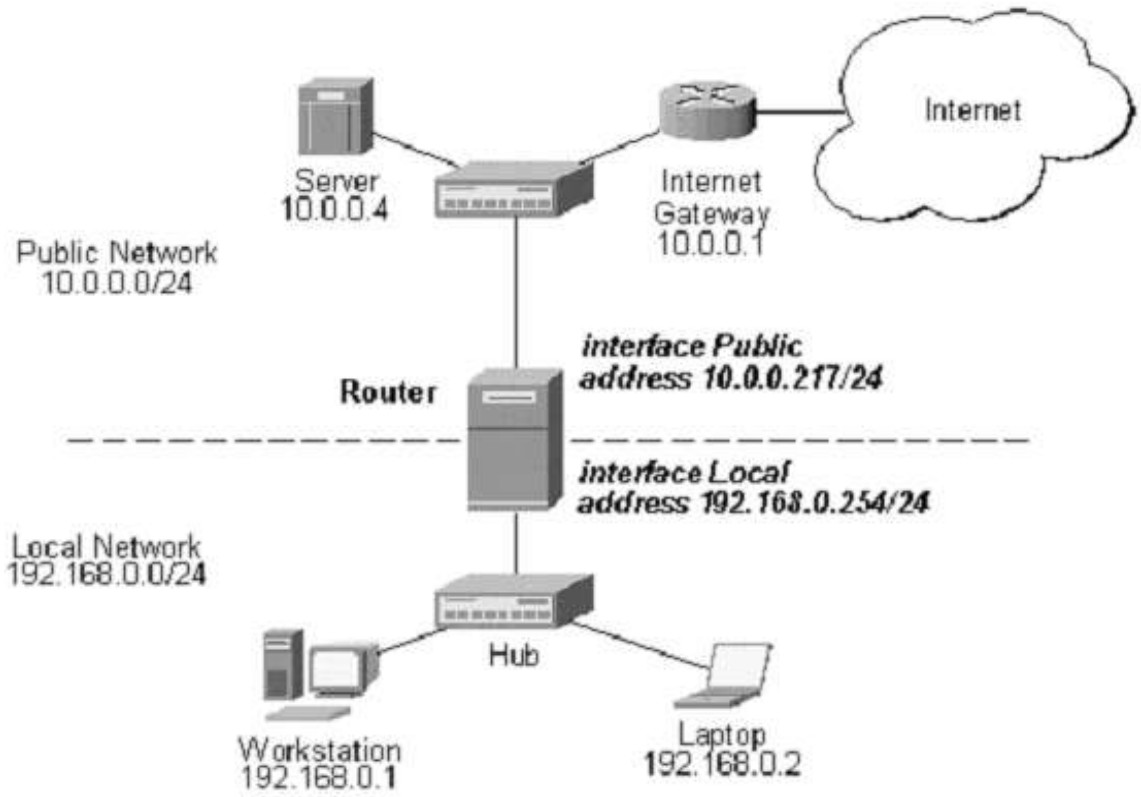
```
 * a - add ip address
 - g - setup default gateway
   x - exit menu
your choice [press Enter to add ip address]: a
```

Choose **a** to add an IP address. At first, setup will ask you for an interface to which the address will be assigned. If the setup offers you an undesirable interface, erase this choice, and press the [Tab] key twice to see all available interfaces. After the interface is chosen, assign IP address and network mask on it:

```
your choice: a
enable interface:
ether1  ether2  wlan1
enable interface: ether1
ip address/netmask: 10.1.0.66/24
#Enabling interface
/interface enable ether1
#Adding IP address
/ip address add address=10.1.0.66/24 interface=ether1 comment="added by
setup"
 + a - add ip address
 * g - setup default gateway
   x - exit menu
your choice: x
```

Basic Examples

Assume you need to configure the HauteRoute router for the following network setup:



In the current example we use two networks:

- The local LAN with network address 192.168.0.0 and 24-bit netmask: 255.255.255.0. The router's address is 192.168.0.254 in this network
- The ISP's network with address 10.0.0.0 and 24-bit netmask 255.255.255.0. The router's address is 10.0.0.217 in this network

The addresses can be added and viewed using the following commands:

```
[admin@HauteRoute] ip address> add address 10.0.0.217/24 interface
Public
[admin@HauteRoute] ip address> add address 192.168.0.254/24 interface
Local
[admin@HauteRoute] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
  #   ADDRESS            NETWORK         BROADCAST       INTERFACE
  0   10.0.0.217/24      10.0.0.217      10.0.0.255      Public
  1   192.168.0.254/24   192.168.0.0     192.168.0.255   Local
[admin@HauteRoute] ip address>
```

Here, the network mask has been specified in the value of the address argument. Alternatively, the argument 'netmask' could have been used with the value '255.255.255.0'. The network and broadcast addresses were not specified in the input since they could be calculated automatically.

Please **note** that the addresses assigned to different interfaces of the router should belong to different networks.

Viewing Routes

You can see two dynamic (D) and connected (C) routes, which have been added automatically when the addresses were added in the example above:

```
[admin@HauteRoute] ip route> print
Flags: X - disabled, I - invalid, D - dynamic, J - rejected,
C - connect, S - static, R - rip, O - ospf, B - bgp
    #    DST-ADDRESS        G GATEWAY         DISTANCE INTERFACE
    0 DC 192.168.0.0/24     r 0.0.0.0         0        Local
    1 DC 10.0.0.0/24        r 0.0.0.0         0        Public
[admin@HauteRoute] ip route> print detail
Flags: X - disabled, I - invalid, D - dynamic, J - rejected,
C - connect, S - static, R - rip, O - ospf, B - bgp
    0 DC dst-address=192.168.0.0/24 preferred-source=192.168.0.254
         gateway=0.0.0.0 gateway-state=reachable distance=0
interface=Local

    1 DC dst-address=10.0.0.0/24 preferred-source=10.0.0.217
gateway=0.0.0.0
         gateway-state=reachable distance=0 interface=Public

[admin@HauteRoute] ip route>
```

These routes show, that IP packets with destination to 10.0.0.0/24 would be sent through the interface Public, whereas IP packets with destination to 192.168.0.0/24 would be sent through the interface Local. However, you need to specify where the router should forward packets, which have destination other than networks connected directly to the router.

Adding Default Routes

In the following example the `default route` (destination 0.0.0.0 (any), netmask 0.0.0.0 (any)) will be added. In this case it is the ISP's gateway 10.0.0.1, which can be reached through the interface **Public**

```
[admin@HauteRoute] ip route> add gateway=10.0.0.1
[admin@HauteRoute] ip route> print
Flags: X - disabled, I - invalid, D - dynamic, J - rejected,
C - connect, S - static, R - rip, O - ospf, B - bgp
    #    DST-ADDRESS        G GATEWAY         DISTANCE INTERFACE
    0  S 0.0.0.0/0          r 10.0.0.1        1        Public
    1 DC 192.168.0.0/24     r 0.0.0.0         0        Local
    2 DC 10.0.0.0/24        r 0.0.0.0         0        Public
[admin@HauteRoute] ip route>
```

Here, the default route is listed under #0. As we see, the gateway 10.0.0.1 can be reached through the interface 'Public'. If the gateway was specified incorrectly, the value for the argument 'interface' would be unknown.

Notes

You cannot add two routes to the same destination, i.e., destination-address/netmask! It applies to the default routes as well. Instead, you can enter multiple gateways for one destination. For

more information on IP routes, please read the Routes, Equal Cost Multipath Routing, Policy Routing manual.

If you have added an unwanted static route accidentally, use the **remove** command to delete the unneeded one. You will not be able to delete dynamic (DC) routes. They are added automatically and represent routes to the networks the router connected directly.

Testing the Network Connectivity

From now on, the **/ping** command can be used to test the network connectivity on both interfaces. You can reach any host on both connected networks from the router.

How the **/ping** command works:

```
[admin@HauteRoute] ip route> /ping 10.0.0.4
10.0.0.4 64 byte ping: ttl=255 time=7 ms
10.0.0.4 64 byte ping: ttl=255 time=5 ms
10.0.0.4 64 byte ping: ttl=255 time=5 ms
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 5/5.6/7 ms
[admin@HauteRoute] ip route>
[admin@HauteRoute] ip route> /ping 192.168.0.1
192.168.0.1 64 byte ping: ttl=255 time=1 ms
192.168.0.1 64 byte ping: ttl=255 time=1 ms
192.168.0.1 64 byte ping: ttl=255 time=1 ms
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 1/1.0/1 ms
[admin@HauteRoute] ip route>
```

The workstation and the laptop can reach (ping) the router at its local address 192.168.0.254, If the router's address 192.168.0.254 is specified as the default gateway in the TCP/IP configuration of both the workstation and the laptop, then you should be able to ping the router:

```
C:\>ping 192.168.0.254
Reply from 192.168.0.254: bytes=32 time=10ms TTL=253
Reply from 192.168.0.254: bytes=32 time<10ms TTL=253
Reply from 192.168.0.254: bytes=32 time<10ms TTL=253

C:\>ping 10.0.0.217
Reply from 10.0.0.217: bytes=32 time=10ms TTL=253
Reply from 10.0.0.217: bytes=32 time<10ms TTL=253
Reply from 10.0.0.217: bytes=32 time<10ms TTL=253

C:\>ping 10.0.0.4
Request timed out.
Request timed out.
Request timed out.
```

Notes

You cannot access anything beyond the router (network 10.0.0.0/24 and the Internet), unless you do the one of the following:

- Use source network address translation (masquerading) on the HauteRoute router to 'hide' your private LAN 192.168.0.0/24 (see the information below), or
- Add a static route on the ISP's gateway 10.0.0.1, which specifies the host 10.0.0.217 as the gateway to network 192.168.0.0/24. Then all hosts on the ISP's network, including the server, will be able to communicate with the hosts on the LAN

To set up routing, it is required that you have some knowledge of configuring TCP/IP networks.

There is a comprehensive list of IP resources compiled by Uri Raz at http://www.private.org.il/tcpip_rl.html. We strongly recommend that you obtain more knowledge, if you have difficulties configuring your network setups.

Advanced Configuration Tasks

Next will be discussed situation with 'hiding' the private LAN 192.168.0.0/24 'behind' one address 10.0.0.217 given to you by the ISP.

Application Example with Masquerading

If you want to 'hide' the private LAN 192.168.0.0/24 'behind' one address 10.0.0.217 given to you by the ISP, you should use the source network address translation (masquerading) feature of the HauteRoute router. Masquerading is useful, if you want to access the ISP's network and the Internet appearing as all requests coming from the host 10.0.0.217 of the ISP's network. The masquerading will change the source IP address and port of the packets originated from the network 192.168.0.0/24 to the address 10.0.0.217 of the router when the packet is routed through it.

Masquerading conserves the number of global IP addresses required and it lets the whole network use a single IP address in its communication with the world.

To use masquerading, a source NAT rule with action 'masquerade' should be added to the firewall configuration:

```
[admin@HauteRoute] ip firewall src-nat> add action=masquerade out-
interface=Public
[admin@HauteRoute] ip firewall src-nat> print
Flags: X - disabled, I - invalid, D - dynamic
 0   out-interface=Public action=masquerade src-address=192.168.0.0/24
[admin@HauteRoute] ip firewall src-nat>
```

Notes

Please consult Network Address Translation Manual for more information on masquerading.

Example with Bandwidth Management

Assume you want to limit the bandwidth to 128kbps on downloads and 64kbps on uploads for all hosts on the LAN. Bandwidth limitation is done by applying queues for outgoing interfaces regarding the traffic flow. It is enough to add a single queue at the HauteRoute router:

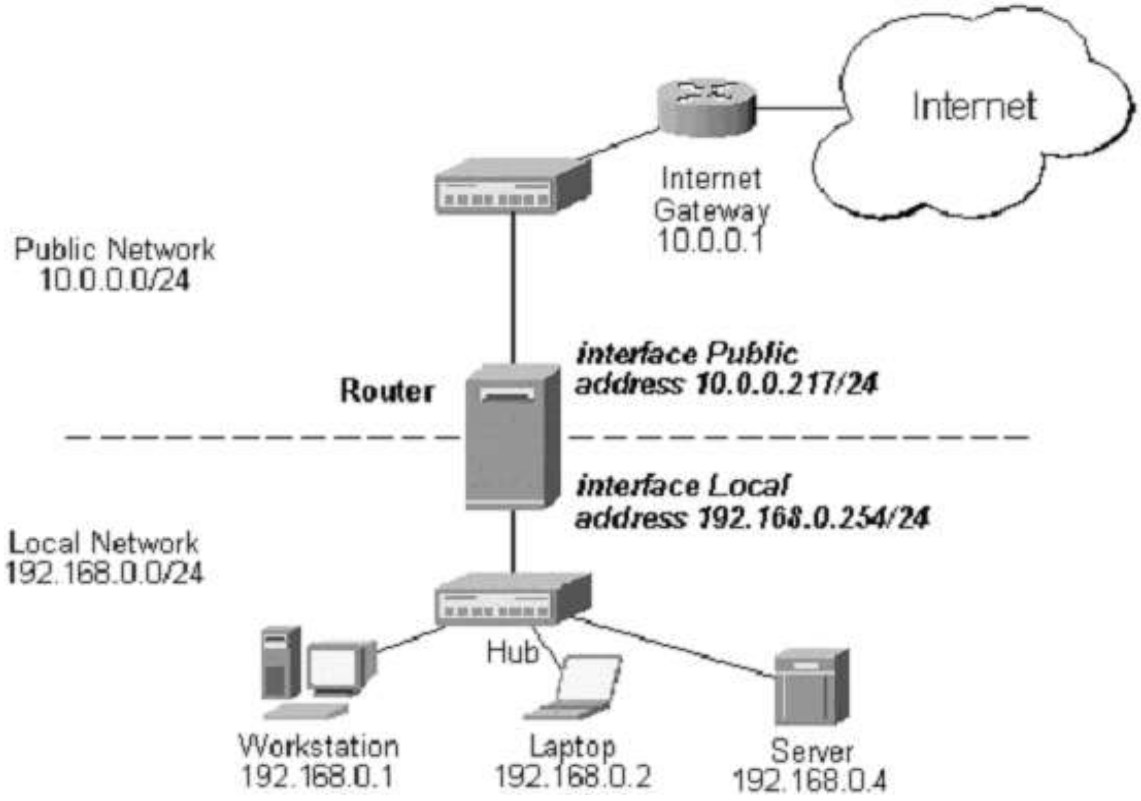
```
[admin@HauteRoute] queue simple> add maX+limit=64000/128000
interface=Local
```

```
[admin@HauteRoute] queue simple> print
Flags: X - disabled, I - invalid, D - dynamic
 0    name="queue1" target-address=0.0.0.0/0 dst-address=0.0.0.0/0
      interface=Local queue=default priority=8 limit-at=0/0
      maX+limit=64000/128000
[admin@HauteRoute] queue simple>
```

Leave all other parameters as set by default. The limit is approximately 128kbps going to the LAN (download) and 64kbps leaving the client's LAN (upload).

Example with NAT

Assume we have moved the server in our previous examples from the public network to our local one:



The server's address is now 192.168.0.4, and we are running web server on it that listens to the TCP port 80. We want to make it accessible from the Internet at address:port 10.0.0.217:80. This can be done by means of Static Network Address translation (NAT) at the HauteRoute Router. The Public address:port 10.0.0.217:80 will be translated to the Local address:port 192.168.0.4:80. One destination NAT rule is required for translating the destination address and port:

```
[admin@HauteRoute] ip firewall dst-nat> add action=nat protocol=tcp \
dst-address=10.0.0.217/32:80 to-dst-address=192.168.0.4
[admin@HauteRoute] ip firewall dst-nat> print
Flags: X - disabled, I - invalid, D - dynamic
 0   dst-address=10.0.0.217/32:80 protocol=tcp action=nat
     to-dst-address=192.168.0.4
```

```
[admin@HauteRoute] ip firewall dst-nat>
```

Notes

Please consult Network Address Translation Manual for more information on Network Address Translation.

Special Notes on the -9 (902-928MHz) Option

The HauteWRAP HR-WRAP series of routers can be ordered with a 900MHz option. This option is implemented using a down converter which makes the router appear as though it is operating at 2.4GHz.

There is a channel for channel down conversion mapping which is important to be aware.

802.11 Channel	802.11 Freq	900MHz Freq	Supported Bandwidth
3	2422	922	5/10 MHz
4	2427	917	5/10/20 MHz
5	2432	912	5/10/20 MHz
6	2437	907	5/10 MHz

Since the 900MHz ISM spectrum is only 20MHz wide in it's entirety, it is generally not reasonable to expect that you will be able to support a channel width of 20MHz reliably. Interference will be much more likely if the entire channel is used. Therefore, we recommend a channel width of 5MHz in most cases. In rural areas, 10 and 20MHz channel widths are possible to support, but we recommend a complete survey of the area before attempting such link speeds.

Channel width and data rate are directly related, along with modulation scheme. We have found that 16-QAM and 64-QAM modulation schemes are less reliable at 900MHz and so we recommend using other, less complex modulation. Practically speaking, the best performance will be had when operating with a 5MHz channel width with a maximum link speed of no more than 24Mbps. This will lock your modulation out of rate scaling up to 16-QAM or 64-QAM.

What this means is that you can really expect performance at 900MHz to never exceed 18Mbps under ideal conditions (20MHz channel width running QPSK modulation) and practically your speed will be more like 5-8Mbps (5Mhz channel width with QPSK or lower modulation).

Our recommended settings for 900MHz are as follows (as configured in the wireless interface dialog) Most relevant settings are highlighted in bold:

mtu=1500
arp=enabled
disable-running-check=no
mode=ap-bridge
ssid="whatever"
area=""
frequency-mode=manual-txpower
country=no_country_set
antenna-gain=0
frequency=2437
band=2ghz-5mhz
scan-list=default
rate-set=configured
supported-rates-b=1Mbps,2Mbps,5.5Mbps,11Mbps
supported-rates-a/g=6Mbps,9Mbps,12Mbps,18Mbps,24Mbps (note that rates over 24 are disabled)
basic-rates-b=1Mbps
basic-rates-a/g=6Mbps
maX+station-count=2007
ack-timeout=dynamic
tX+power=28
tX+power-mode=card-rates
noise-floor-threshold=default
periodic-calibration=default
periodic-calibration-interval=60
burst-time=disabled dfs-mode=none
antenna-mode=ant-a
wds-mode=enabled
wds-default-bridge=bridge1
wds-default-cost=100
wds-cost-range=50-150
wds-ignore-ssid=no
update-stats-interval=disabled
default-authentication=yes
default-forwarding=yes
default-ap-tX+limit=0

default-client-tX+limit=0
hide-ssid=no
security-profile=default
disconnect-timeout=3s
on-fail-retry-time=100ms
preamble-mode=both compression=no
allow-sharedkey=no

Instructions for configuring a point to point link as shown in previous chapters should then be used.

Federal Communications Commission Statement

This device contains one or more of the following modular components which have been tested and approved as “FCC approved modules”.

FCC ID for each module is as follows:
-M option - TV7R52-350
-N option - TV7R52HN
-2 option – NI3-86005001
-2a option – SWX+SR2
-2x option - SWX+XR2
-4 option – SWX+SR4
-5 option – SWX+SR5
-5x option – SWX+XR5
-9 option – SWX+SR9
-9x option – SWX+XR9
WRAPterBASE5 - TV7RB711GA-5HND
WRAPterSTATION5 - TV7RB711G-5HND
LINK101 - TV7RB751-2HND

1. Each modular transmitter has its own RF shielding (the shielding must enclose all RF circuitry).

2. Each modular transmitter has buffered modulation/data inputs.

3. Each modular transmitter has its own power supply regulation to ensure that the module will comply with Part 15 requirements regardless of the design of the power supplying circuitry for the host. (U. S.)

4. The certification submission shall contain a detailed description of the configuration of all antennas that will be used with the module. All modules are approved for use with 4dBi gain rubber duck type omni directional antennas.

5. Each modular transmitter has a "unique" antenna coupler (at all connections between the module and the antenna, including the cable). The "professional installation" provision of Section 15.203 may not be applied to modules. (U. S.)

6. Each modular transmitter has been tested in a stand-alone configuration. (U.S.)

7. Unless battery powered, each module complies with the AC line conducted requirements found in Section 15.207 (U. S.)

8. Each module meets Industry Canada's certification labeling requirements. And each host device shall also comply with the certification labeling requirements of each of the modules it contains. (Canada)

9. Each modular transmitter has been labeled with its own FCC ID number and the label is visible form the outside of the host device. Each host device also displays a label referring to the enclosed module's FCC ID. (U. S.)

10. Each modular transmitter complies with any specific rule or operating requirements applicable to the transmitter and the manufacturer must provide adequate instructions with the module to explain any such requirements.

11. The host device and all the separately certified modules it contains jointly meet the RF safety requirements of RSS-102, if applicable. (Canada)

12. Each modular transmitter complies with any applicable FCC RF exposure requirements. (U.S.)

Changes or modifications not expressly approved by the manufacturer could void your right to operate the equipment.

Test reports for each module and emissions test reports for the host computer board are available on request.

CE Declaration of Conformity

We, HauteSpot Networks Corporation, declares that the HR-IXP series of routers, is in conformance with:
- EN 61000-6-3 and EN 61000-6-4 (EMI emissions, residential and industrial)
- EN 61000-6-1 and EN 61000-6-2 (ESD, susceptibility, residential and industrial)

A copy of the test report will be provided on request.

Technical data

Radio Module Options:

- 2 Option (2.4 GHz)

- Chipset: Atheros AR5006xs
- Compliant: FCC Part15/UL/ETSI 300/328/CE
- Media Access Protocol: CSMA/CA
- Modulation Technology: Orthogonal Frequency Division Multiplexing(OFDM)
  - DBPSK @ 1Mbps
  - DQPSK @ 2Mbps
  - CCK @ 5.5 & 11Mbps
  - BPSK @ 6&9Mbps
  - QPSK @ 12&18Mbps
  - 16-QAM @ 24&36Mbps
  - 64-QAM @ 48&54Mbps
- Frequency Band:2.312~2.499 GHz
- Radio Type: Direct Sequence Spread Spectrum (DSSS)
- Operation Channels: 11 for North America
- Current Consumption:
  - Continue Tx: </= 750mA
  - Continue Rx: </= 400mA
  - LINK Mode: </= 400mA
- Form Factor: Mini-PCI Type IIIA
- Dimensions: 59.60mm x 44.45mm
- Antenna Connector: 2x U.FL Connectors
- Operating Voltage: 5V +/- 0.15V (Input: 3.3V)
- Operating Temp: 0ºC to 55ºC
- Storage Temp: -20ºC to 70ºC
- Operating Humidity: 5%~95% Typical

Radio	Devices	Speed Mbps	Modulation	Transmit Output Power dBm	Transmit Output Power mW	Receive Sensitivity dBm	Frequency MHz	IEEE Specs
Atheros AR5004	-2 option	54	64-QAM	21	126	-74	2400-2484	802.11b/g
Atheros AR5004	-2 option	48	64-QAM	22	158	-77	2400-2484	802.11b/g
Atheros AR5004	-2 option	36	16-QAM	24	251	-83	2400-2484	802.11b/g
Atheros AR5004	-2 option	24	16-QAM	26	398	-86	2400-2484	802.11b/g
Atheros AR5004	-2 option	18	QPSK	26	398	-90	2400-2484	802.11b/g
Atheros AR5004	-2 option	12	QPSK	26	398	-91	2400-2484	802.11b/g
Atheros AR5004	-2 option	11	CCK	26	398	-92	2400-2484	802.11b/g
Atheros AR5004	-2 option	9	BPSK	26	398	-93	2400-2484	802.11b/g
Atheros AR5004	-2 option	6	BPSK	26	398	-94	2400-2484	802.11b/g
Atheros AR5004	-2 option	5.5	CCK	26	398	-95	2400-2484	802.11b/g
Atheros AR5004	-2 option	2	DQPSK	26	398	-96	2400-2484	802.11b/g
Atheros AR5004	-2 option	1	DQPSK	26	398	-97	2400-2484	802.11b/g

- 2a Option (2.4 GHz) Multi Band

- Chipset: Atheros AR5006xs
- Compliant: FCC Part15/UL/ETSI 300/328/CE
- Media Access Protocol: CSMA/CA
- Modulation Technology: Orthogonal Frequency Division Multiplexing(OFDM)
  - DBPSK @ 1Mbps
  - DQPSK @ 2Mbps
  - CCK @ 5.5 & 11Mbps
  - BPSK @ 6&9Mbps
  - QPSK @ 12&18Mbps

16-QAM @ 24&36Mbps
64-QAM @ 48&54Mbps

- Frequency Band:2.312~2.499 GHz
- Radio Type: Direct Sequence Spread Spectrum (DSSS)
- Operation Channels: 11 for North America
- Current Consumption:
  - Continue Tx: </= 750mA
  - Continue Rx: </= 400mA
  - LINK Mode: </= 400mA
- Form Factor: Mini-PCI Type IIIA
- Dimensions: 59.60mm x 44.45mm
- Antenna Connector: 2x U.FL Connectors
- Operating Voltage: 5V +/- 0.15V (Input: 3.3V)
- Operating Temp: 0ºC to 55ºC
- Storage Temp: -20ºC to 70ºC
- Operating Humidity: 5%~95% Typical

Radio	Devices	Speed Mbps	Modulation	Transmit Output Power dBm	Transmit Output Power mW	Receive Sensitivity dBm	Frequency MHz	IEEE Specs
Atheros AR5006	-2a Option	54	64-QAM	23	200	-76	2400-2484	802.11b/g
Atheros AR5006	-2a Option	48	64-QAM	24	251	-78	2400-2484	802.11b/g
Atheros AR5006	-2a Option	36	16-QAM	25	316	-80	2400-2484	802.11b/g
Atheros AR5006	-2a Option	24	16-QAM	26	398	-85	2400-2484	802.11b/g
Atheros AR5006	-2a Option	18	QPSK	26	398	-87	2400-2484	802.11b/g
Atheros AR5006	-2a Option	12	QPSK	26	398	-90	2400-2484	802.11b/g
Atheros AR5006	-2a Option	11	CCK	26	398	-92	2400-2484	802.11b/g
Atheros AR5006	-2a Option	9	BPSK	26	398	-91	2400-2484	802.11b/g
Atheros AR5006	-2a Option	6	BPSK	26	398	-92	2400-2484	802.11b/g
Atheros AR5006	-2a Option	5.5	CCK	26	398	-96	2400-2484	802.11b/g
Atheros AR5006	-2a Option	2	DQPSK	26	398	-96	2400-2484	802.11b/g
Atheros AR5006	-2a Option	1	DQPSK	26	398	-96	2400-2484	802.11b/g
Atheros AR5006	-2a Option	54	64-QAM	17	50	-74	5100-5900	802.11a
Atheros AR5006	-2a Option	48	64-QAM	18	63	-77	5100-5900	802.11a
Atheros AR5006	-2a Option	36	16-QAM	19	79	-79	5100-5900	802.11a
Atheros AR5006	-2a Option	24	16-QAM	20	100	-84	5100-5900	802.11a
Atheros AR5006	-2a Option	18	QPSK	20	100	-87	5100-5900	802.11a
Atheros AR5006	-2a Option	12	QPSK	20	100	-88	5100-5900	802.11a
Atheros AR5006	-2a Option	9	BPSK	20	100	-89	5100-5900	802.11a
Atheros AR5006	-2a Option	6	BPSK	20	100	-90	5100-5900	802.11a

- 2x Option (2.4 GHz) High Power

- Chipset: Atheros AR5006xs
- Compliant: FCC Part15/UL/ETSI 300/328/CE
- Media Access Protocol: CSMA/CA
- Modulation Technology: Orthogonal Frequency Division Multiplexing(OFDM)
  DBPSK @ 1Mbps
  DQPSK @ 2Mbps
  CCK @ 5.5 & 11Mbps
  BPSK @ 6&9Mbps
  QPSK @ 12&18Mbps
  16-QAM @ 24&36Mbps
  64-QAM @ 48&54Mbps
- Frequency Band:2.312~2.499 GHz
- Radio Type: Direct Sequence Spread Spectrum (DSSS)
- Operation Channels: 11 for North America
- Current Consumption:

    - Continue Tx: </= 750mA
    - Continue Rx: </= 400mA
    - LINK Mode: </= 400mA
- Form Factor: Mini-PCI Type IIIA
- Dimensions: 59.60mm x 44.45mm
- Antenna Connector: 2x U.FL Connectors
- Operating Voltage: 5V +/- 0.15V (Input: 3.3V)
- Operating Temp: 0ºC to 55ºC
- Storage Temp: -20ºC to 70ºC
- Operating Humidity: 5%~95% Typical

Radio	Devices	Speed Mbps	Modulation	Transmit Output Power dBm	Transmit Output Power mW	Receive Sensitivity dBm	Frequency MHz	IEEE Specs
Atheros AR5006	-2x option	54	64-QAM	24	251	-74	2400-2484	802.11b/g
Atheros AR5006	-2x option	48	64-QAM	25	316	-77	2400-2484	802.11b/g
Atheros AR5006	-2x option	36	16-QAM	26	398	-83	2400-2484	802.11b/g
Atheros AR5006	-2x option	24	16-QAM	28	631	-86	2400-2484	802.11b/g
Atheros AR5006	-2x option	18	QPSK	28	631	-90	2400-2484	802.11b/g
Atheros AR5006	-2x option	12	QPSK	28	631	-91	2400-2484	802.11b/g
Atheros AR5006	-2x option	11	CCK	28	631	-92	2400-2484	802.11b/g
Atheros AR5006	-2x option	9	BPSK	28	631	-93	2400-2484	802.11b/g
Atheros AR5006	-2x option	6	BPSK	28	631	-94	2400-2484	802.11b/g
Atheros AR5006	-2x option	5.5	CCK	28	631	-95	2400-2484	802.11b/g
Atheros AR5006	-2x option	2	DQPSK	28	631	-96	2400-2484	802.11b/g
Atheros AR5006	-2x option	1	DQPSK	28	631	-97	2400-2484	802.11b/g

- 3a Option (3.3 GHz)

- Chipset: Atheros AR5006xs
- Compliant: Approval Required, see local regulations
- Media Access Protocol: CSMA/CA
- Modulation Technology: Orthogonal Frequency Division Multiplexing(OFDM)
    DBPSK @ 1Mbps
    DQPSK @ 2Mbps
    CCK @ 5.5 & 11Mbps
    BPSK @ 6&9Mbps
    QPSK @ 12&18Mbps
    16-QAM @ 24&36Mbps
    64-QAM @ 48&54Mbps
- Frequency Band:3300~3450 MHz
- Radio Type: Direct Sequence Spread Spectrum (DSSS)
- Operation Channels: 11 for North America
- Current Consumption:
    - Continue Tx: </= 750mA
    - Continue Rx: </= 400mA
    - LINK Mode: </= 400mA
- Form Factor: Mini-PCI Type IIIA
- Dimensions: 59.60mm x 44.45mm
- Antenna Connector: 2x U.FL Connectors
- Operating Voltage: 5V +/- 0.15V (Input: 3.3V)
- Operating Temp: 0ºC to 55ºC
- Storage Temp: -20ºC to 70ºC
- Operating Humidity: 5%~95% Typical

Radio	Devices	Speed Mbps	Modulation	Transmit Output Power dBm	Transmit Output Power mW	Receive Sensitivity dBm	Frequency MHz	IEEE Specs
Atheros AR5006	-3a option	54	64-QAM	18	63	-73	3300-3450	Proprietary OFDM
Atheros AR5006	-3a option	48	64-QAM	19	79	-75	3300-3450	Proprietary OFDM
Atheros AR5006	-3a option	36	16-QAM	21	126	-81	3300-3450	Proprietary OFDM
Atheros AR5006	-3a option	24	16-QAM	25	316	-84	3300-3450	Proprietary OFDM
Atheros AR5006	-3a option	18	QPSK	25	316	-88	3300-3450	Proprietary OFDM
Atheros AR5006	-3a option	12	QPSK	25	316	-89	3300-3450	Proprietary OFDM
Atheros AR5006	-3a option	9	BPSK	25	316	-90	3300-3450	Proprietary OFDM
Atheros AR5006	-3a option	6	BPSK	25	316	-92	3300-3450	Proprietary OFDM

- 3b Option (3.475 GHz)

- Chipset: Atheros AR5006xs
- Compliant: Approval Required, see local regulations
- Media Access Protocol: CSMA/CA
- Modulation Technology: Orthogonal Frequency Division Multiplexing(OFDM)
  - DBPSK @ 1Mbps
  - DQPSK @ 2Mbps
  - CCK @ 5.5 & 11Mbps
  - BPSK @ 6&9Mbps
  - QPSK @ 12&18Mbps
  - 16-QAM @ 24&36Mbps
  - 64-QAM @ 48&54Mbps
- Frequency Band: 3475~3625 MHz
- Radio Type: Direct Sequence Spread Spectrum (DSSS)
- Operation Channels: 11 for North America
- Current Consumption:
  - Continue Tx: </= 750mA
  - Continue Rx: </= 400mA
  - LINK Mode: </= 400mA
- Form Factor: Mini-PCI Type IIIA
- Dimensions: 59.60mm x 44.45mm
- Antenna Connector: 2x U.FL Connectors
- Operating Voltage: 5V +/- 0.15V (Input: 3.3V)
- Operating Temp: 0ºC to 55ºC
- Storage Temp: -20ºC to 70ºC
- Operating Humidity: 5%~95% Typical

Radio	Devices	Speed Mbps	Modulation	Transmit Output Power dBm	Transmit Output Power mW	Receive Sensitivity dBm	Frequency MHz	IEEE Specs
Atheros AR5006	-3b option	54	64-QAM	18	63	-73	3475-3625	Proprietary OFDM
Atheros AR5006	-3b option	48	64-QAM	19	79	-75	3475-3625	Proprietary OFDM
Atheros AR5006	-3b option	36	16-QAM	21	126	-81	3475-3625	Proprietary OFDM
Atheros AR5006	-3b option	24	16-QAM	25	316	-84	3475-3625	Proprietary OFDM
Atheros AR5006	-3b option	18	QPSK	25	316	-88	3475-3625	Proprietary OFDM
Atheros AR5006	-3b option	12	QPSK	25	316	-89	3475-3625	Proprietary OFDM
Atheros AR5006	-3b option	9	BPSK	25	316	-90	3475-3625	Proprietary OFDM
Atheros AR5006	-3b option	6	BPSK	25	316	-92	3475-3625	Proprietary OFDM

- 3c Option (3.6 GHz)

- Chipset: Atheros AR5006xs
- Compliant: Approval Required, see local regulations
- Media Access Protocol: CSMA/CA
- Modulation Technology: Orthogonal Frequency Division Multiplexing(OFDM)

DBPSK @ 1Mbps
DQPSK @ 2Mbps
CCK @ 5.5 & 11Mbps
BPSK @ 6&9Mbps
QPSK @ 12&18Mbps
16-QAM @ 24&36Mbps
64-QAM @ 48&54Mbps

- Frequency Band:3600~3750 MHz
- Radio Type: Direct Sequence Spread Spectrum (DSSS)
- Operation Channels: 11 for North America
- Current Consumption:
  - Continue Tx: </= 750mA
  - Continue Rx: </= 400mA
  - LINK Mode: </= 400mA
- Form Factor: Mini-PCI Type IIIA
- Dimensions: 59.60mm x 44.45mm
- Antenna Connector: 2x U.FL Connectors
- Operating Voltage: 5V +/- 0.15V (Input: 3.3V)
- Operating Temp: 0ºC to 55ºC
- Storage Temp: -20ºC to 70ºC
- Operating Humidity: 5%~95% Typical

Radio	Devices	Speed Mbps	Modulation	Transmit Output Power dBm	Transmit Output Power mW	Receive Sensitivity dBm	Frequency MHz	IEEE Specs
Atheros AR5006	-3c option	54	64-QAM	18	63	-73	3600-3750	Proprietary OFDM
Atheros AR5006	-3c option	48	64-QAM	19	79	-75	3600-3750	Proprietary OFDM
Atheros AR5006	-3c option	36	16-QAM	21	126	-81	3600-3750	Proprietary OFDM
Atheros AR5006	-3c option	24	16-QAM	25	316	-84	3600-3750	Proprietary OFDM
Atheros AR5006	-3c option	18	QPSK	25	316	-88	3600-3750	Proprietary OFDM
Atheros AR5006	-3c option	12	QPSK	25	316	-89	3600-3750	Proprietary OFDM
Atheros AR5006	-3c option	9	BPSK	25	316	-90	3600-3750	Proprietary OFDM
Atheros AR5006	-3c option	6	BPSK	25	316	-92	3600-3750	Proprietary OFDM

- 4 Option (4.9GHz)

- Chipset: Atheros AR5006xs
- Compliant: FCC Part90Y Part 15-247/UL/ETSI 300/328/CE
- Media Access Protocol: CSMA/CA
- Modulation Technology: Orthogonal Frequency Division Multiplexing(OFDM)

DBPSK @ 1Mbps
DQPSK @ 2Mbps
CCK @ 5.5 & 11Mbps
BPSK @ 6&9Mbps
QPSK @ 12&18Mbps
16-QAM @ 24&36Mbps
64-QAM @ 48&54Mbps

- Frequency Band:4.900-4.990 GHz
- Radio Type: Direct Sequence Spread Spectrum (DSSS)
- Current Consumption:
  - Continue Tx: </= 800mA
  - Continue Rx: </= 400mA
  - LINK Mode: </= 400mA
- Form Factor: Mini-PCI Type IIIA
- Dimensions: 59.60mm x 44.45mm

- Antenna Connector: 2x U.FL Connectors
- Operating Voltage: 5V +/- 0.15V (Input: 3.3V)
- Operating Temp: -20ºC to 60ºC
- Storage Temp: -70ºC to 85ºC
- Operating Humidity: 5%~95% Typical

Radio	Devices	Speed Mbps	Modulation	Transmit Output Power dBm	Transmit Output Power mW	Receive Sensitivity dBm	Frequency MHz	IEEE Specs
Atheros AR5004	-4 Option	54	64-QAM	21	126	-74	4940-4990	FCC Part 90Y
Atheros AR5004	-4 Option	48	64-QAM	22	158	-77	4940-4990	FCC Part 90Y
Atheros AR5004	-4 Option	36	16-QAM	24	251	-83	4940-4990	FCC Part 90Y
Atheros AR5004	-4 Option	24	16-QAM	26	398	-86	4940-4990	FCC Part 90Y
Atheros AR5004	-4 Option	18	QPSK	26	398	-90	4940-4990	FCC Part 90Y
Atheros AR5004	-4 Option	12	QPSK	26	398	-91	4940-4990	FCC Part 90Y
Atheros AR5004	-4 Option	9	BPSK	26	398	-93	4940-4990	FCC Part 90Y
Atheros AR5004	-4 Option	6	BPSK	26	398	-94	4940-4990	FCC Part 90Y

- 5 Option (5.1-5.8GHz)

- Chipset: Atheros AR5004
- Compliant: FCC Part 15/UL/ETSI 300/328/CE
- Media Access Protocol: CSMA/CA
- Modulation Technology: Orthogonal Frequency Division Multiplexing(OFDM)
  - DBPSK @ 1Mbps
  - DQPSK @ 2Mbps
  - CCK @ 5.5 & 11Mbps
  - BPSK @ 6&9Mbps
  - QPSK @ 12&18Mbps
  - 16-QAM @ 24&36Mbps
  - 64-QAM @ 48&54Mbps
- Frequency Band:5.100-5.900 GHz
- Radio Type: Direct Sequence Spread Spectrum (DSSS)
- Current Consumption:
  - Continue Tx: </= 800mA
  - Continue Rx: </= 400mA
  - LINK Mode: </= 400mA
- Form Factor: Mini-PCI Type IIIA
- Dimensions: 59.60mm x 44.45mm
- Antenna Connector: 2x U.FL Connectors
- Operating Voltage: 5V +/- 0.15V (Input: 3.3V)
- Operating Temp: -40ºC to 60ºC
- Storage Temp: -70ºC to 85ºC
- Operating Humidity: 5%~95% Typical

Radio	Devices	Speed Mbps	Modulation	Transmit Output Power dBm	Transmit Output Power mW	Receive Sensitivity dBm	Frequency MHz	IEEE Specs
Atheros AR5004	-5 Option	54	64-QAM	21	126	-74	5100-5900	802.11a
Atheros AR5004	-5 Option	48	64-QAM	22	158	-77	5100-5900	802.11a
Atheros AR5004	-5 Option	36	16-QAM	24	251	-83	5100-5900	802.11a
Atheros AR5004	-5 Option	24	16-QAM	26	398	-86	5100-5900	802.11a
Atheros AR5004	-5 Option	18	QPSK	26	398	-90	5100-5900	802.11a
Atheros AR5004	-5 Option	12	QPSK	26	398	-91	5100-5900	802.11a
Atheros AR5004	-5 Option	9	BPSK	26	398	-93	5100-5900	802.11a
Atheros AR5004	-5 Option	6	BPSK	26	398	-94	5100-5900	802.11a

- 5x Option (5.1-5.8GHz) High Power

- Chipset: Atheros AR5004
- Compliant: FCC Part 15/UL/ETSI 300/328/CE
- Media Access Protocol: CSMA/CA
- Modulation Technology: Orthogonal Frequency Division Multiplexing(OFDM)
  - DBPSK @ 1Mbps
  - DQPSK @ 2Mbps
  - CCK @ 5.5 & 11Mbps
  - BPSK @ 6&9Mbps
  - QPSK @ 12&18Mbps
  - 16-QAM @ 24&36Mbps
  - 64-QAM @ 48&54Mbps
- Frequency Band:5.100-5.900 GHz
- Radio Type: Direct Sequence Spread Spectrum (DSSS)
- Current Consumption:
  - Continue Tx: </= 800mA
  - Continue Rx: </= 400mA
  - LINK Mode: </= 400mA
- Form Factor: Mini-PCI Type IIIA
- Dimensions: 59.60mm x 44.45mm
- Antenna Connector: 2x U.FL Connectors
- Operating Voltage: 5V +/- 0.15V (Input: 3.3V)
- Operating Temp: -40ºC to 60ºC
- Storage Temp: -70ºC to 85ºC
- Operating Humidity: 5%~95% Typical

Radio	Devices	Speed Mbps	Modulation	Transmit Output Power dBm	Transmit Output Power mW	Receive Sensitivity dBm	Frequency MHz	IEEE Specs
Atheros AR5006	-5x Option	54	64-QAM	23	200	-74	5100-5900	802.11a
Atheros AR5006	-5x Option	48	64-QAM	24	251	-77	5100-5900	802.11a
Atheros AR5006	-5x Option	36	16-QAM	26	398	-83	5100-5900	802.11a
Atheros AR5006	-5x Option	24	16-QAM	28	631	-86	5100-5900	802.11a
Atheros AR5006	-5x Option	18	QPSK	28	631	-90	5100-5900	802.11a
Atheros AR5006	-5x Option	12	QPSK	28	631	-91	5100-5900	802.11a
Atheros AR5006	-5x Option	9	BPSK	28	631	-93	5100-5900	802.11a
Atheros AR5006	-5x Option	6	BPSK	28	631	-94	5100-5900	802.11a

- 7 Option (700MHz)

- Chipset: Atheros AR5213
- Compliant: FCC AWS Licensed Operation Pending
- Media Access Protocol: CSMA/CA
- Modulation Technology: Orthogonal Frequency Division Multiplexing(OFDM)
  - DBPSK @ 1Mbps
  - DQPSK @ 2Mbps
  - CCK @ 5.5 & 11Mbps
  - BPSK @ 6&9Mbps
  - QPSK @ 12&18Mbps
  - 16-QAM @ 24&36Mbps
  - 64-QAM @ 48&54Mbps
- Frequency Band: 746-794 MHz
- Radio Type: Direct Sequence Spread Spectrum (DSSS)

- Current Consumption:
  - Continue Tx: </= 1000mA
  - Continue Rx: </= 400mA
  - LINK Mode: </= 400mA
- Form Factor: Mini-PCI Type IIIA
- Dimensions: 59.60mm x 44.45mm
- Antenna Connector: 2x U.FL Connectors
- Operating Voltage: 5V +/- 0.15V (Input: 3.3V)
- Operating Temp: -40ºC to 60ºC
- Storage Temp: -70ºC to 85ºC
- Operating Humidity: 5%~95% Typical

Radio	Devices	Speed Mbps	Modulation	Transmit Output Power dBm	Transmit Output Power mW	Receive Sensitivity dBm	Frequency MHz	IEEE Specs
Atheros AR5004	-7 Option	54	64-QAM	23	200	-70	746-794	Proprietary OFDM
Atheros AR5004	-7 Option	48	64-QAM	24	251	-73	746-794	Proprietary OFDM
Atheros AR5004	-7 Option	36	16-QAM	26	398	-79	746-794	Proprietary OFDM
Atheros AR5004	-7 Option	24	16-QAM	28	631	-82	746-794	Proprietary OFDM
Atheros AR5004	-7 Option	18	QPSK	28	631	-86	746-794	Proprietary OFDM
Atheros AR5004	-7 Option	12	QPSK	28	631	-87	746-794	Proprietary OFDM
Atheros AR5004	-7 Option	11	CCK	28	631	-88	746-794	Proprietary OFDM
Atheros AR5004	-7 Option	9	BPSK	28	631	-89	746-794	Proprietary OFDM
Atheros AR5004	-7 Option	6	BPSK	28	631	-90	746-794	Proprietary OFDM
Atheros AR5004	-7 Option	5.5	CCK	28	631	-91	746-794	Proprietary OFDM
Atheros AR5004	-7 Option	2	DQPSK	28	631	-92	746-794	Proprietary OFDM
Atheros AR5004	-7 Option	1	DQPSK	28	631	-93	746-794	Proprietary OFDM

- 9 Option (900MHz)

- Chipset: Atheros AR5213
- Compliant: FCC Part 15/UL/ETSI 300/328/CE
- Media Access Protocol: CSMA/CA
- Modulation Technology: Orthogonal Frequency Division Multiplexing(OFDM)
  - DBPSK @ 1Mbps
  - DQPSK @ 2Mbps
  - CCK @ 5.5 & 11Mbps
  - BPSK @ 6&9Mbps
  - QPSK @ 12&18Mbps
  - 16-QAM @ 24&36Mbps
  - 64-QAM @ 48&54Mbps
- Frequency Band: 902-928 MHz
- Radio Type: Direct Sequence Spread Spectrum (DSSS)
- Current Consumption:
  - Continue Tx: </= 1000mA
  - Continue Rx: </= 400mA
  - LINK Mode: </= 400mA
- Form Factor: Mini-PCI Type IIIA
- Dimensions: 59.60mm x 44.45mm
- Antenna Connector: 2x U.FL Connectors
- Operating Voltage: 5V +/- 0.15V (Input: 3.3V)
- Operating Temp: -40ºC to 60ºC
- Storage Temp: -70ºC to 85ºC
- Operating Humidity: 5%~95% Typical

Radio	Devices	Speed Mbps	Modulation	Transmit Output Power dBm	Transmit Output Power mW	Receive Sensitivity dBm	Frequency MHz	IEEE Specs
Atheros AR5004	-9 Option	54	64-QAM	23	200	-70	900-928	Proprietary OFDM
Atheros AR5004	-9 Option	48	64-QAM	24	251	-73	900-928	Proprietary OFDM
Atheros AR5004	-9 Option	36	16-QAM	26	398	-79	900-928	Proprietary OFDM
Atheros AR5004	-9 Option	24	16-QAM	28	631	-82	900-928	Proprietary OFDM
Atheros AR5004	-9 Option	18	QPSK	28	631	-86	900-928	Proprietary OFDM
Atheros AR5004	-9 Option	12	QPSK	28	631	-87	900-928	Proprietary OFDM
Atheros AR5004	-9 Option	11	CCK	28	631	-88	900-928	Proprietary OFDM
Atheros AR5004	-9 Option	9	BPSK	28	631	-89	900-928	Proprietary OFDM
Atheros AR5004	-9 Option	6	BPSK	28	631	-90	900-928	Proprietary OFDM
Atheros AR5004	-9 Option	5.5	CCK	28	631	-91	900-928	Proprietary OFDM
Atheros AR5004	-9 Option	2	DQPSK	28	631	-92	900-928	Proprietary OFDM
Atheros AR5004	-9 Option	1	DQPSK	28	631	-93	900-928	Proprietary OFDM

- N Option (MIMO Multiband)

- Chipset: Atheros AR9220
- Compliant: FCC Part90Y Part 15-247/UL/ETSI 300/328/CE
- Media Access Protocol: CSMA/CA
- Modulation Technology: Orthogonal Frequency Division Multiplexing(OFDM)
  - DBPSK @ 1Mbps
  - DQPSK @ 2Mbps
  - CCK @ 5.5 & 11Mbps
  - BPSK @ 6&9Mbps
  - QPSK @ 12&18Mbps
  - 16-QAM @ 24&36Mbps
  - 64-QAM @ 48&54Mbps
  - 2 x 2 MIMO with rates up to 300Mbps
- Frequency Band:2.192-2.539 GHz and 4.800-6.100 GHz
- Radio Type: Direct Sequence Spread Spectrum (DSSS)
- Current Consumption:
  - Continue Tx: </= 800mA
  - Continue Rx: </= 400mA
  - LINK Mode: </= 400mA
- Form Factor: Mini-PCI Type IIIA
- Dimensions: 59.60mm x 44.45mm
- Antenna Connector: 2x MMCX Connectors
- Operating Voltage: 5V +/- 0.15V (Input: 3.3V)
- Operating Temp: -20ºC to 70ºC
- Storage Temp: -70ºC to 85ºC
- Operating Humidity: 5%~95% Typical

Radio	Devices	Speed Mbps	Modulation	Transmit Output Power dBm	Transmit Output Power mW	Receive Sensitivity dBm	Frequency MHz	IEEE Specs	Notes
Atheros AR9220	-N Option	54	64-QAM	22	158	-81	2192-2539	802.11b/g/n	Unlicensed ISM, support MIMO 802.11n
Atheros AR9220	-N Option	48	64-QAM	22	158	-83	2192-2539	802.11b/g/n	Unlicensed ISM, support MIMO 802.11n
Atheros AR9220	-N Option	36	16-QAM	24	251	-86	2192-2539	802.11b/g/n	Unlicensed ISM, support MIMO 802.11n
Atheros AR9220	-N Option	24	16-QAM	25	316	-90	2192-2539	802.11b/g/n	Unlicensed ISM, support MIMO 802.11n
Atheros AR9220	-N Option	18	QPSK	25	316	-92	2192-2539	802.11b/g/n	Unlicensed ISM, support MIMO 802.11n
Atheros AR9220	-N Option	12	QPSK	25	316	-92	2192-2539	802.11b/g/n	Unlicensed ISM, support MIMO 802.11n
Atheros AR9220	-N Option	11	CCK	24	251	-93	2192-2539	802.11b/g/n	Unlicensed ISM, support MIMO 802.11n
Atheros AR9220	-N Option	9	BPSK	25	316	-94	2192-2539	802.11b/g/n	Unlicensed ISM, support MIMO 802.11n
Atheros AR9220	-N Option	6	BPSK	25	316	-94	2192-2539	802.11b/g/n	Unlicensed ISM, support MIMO 802.11n
Atheros AR9220	-N Option	5.5	CCK	24	251	-93	2192-2539	802.11b/g/n	Unlicensed ISM, support MIMO 802.11n
Atheros AR9220	-N Option	2	DQPSK	24	251	-93	2192-2539	802.11b/g/n	Unlicensed ISM, support MIMO 802.11n
Atheros AR9220	-N Option	1	DQPSK	24	251	-93	2192-2539	802.11b/g/n	Unlicensed ISM, support MIMO 802.11n
Atheros AR9220	-N Option	54	64-QAM	21	126	-80	4800-6100	802.11a/n	Unlicensed ISM, support MIMO 802.11n
Atheros AR9220	-N Option	48	64-QAM	20	100	-83	4800-6100	802.11a/n	Unlicensed ISM, support MIMO 802.11n
Atheros AR9220	-N Option	36	16-QAM	21	126	-86	4800-6100	802.11a/n	Unlicensed ISM, support MIMO 802.11n
Atheros AR9220	-N Option	24	16-QAM	21	126	-88	4800-6100	802.11a/n	Unlicensed ISM, support MIMO 802.11n
Atheros AR9220	-N Option	18	QPSK	24	251	-90	4800-6100	802.11a/n	Unlicensed ISM, support MIMO 802.11n
Atheros AR9220	-N Option	12	QPSK	24	251	-93	4800-6100	802.11a/n	Unlicensed ISM, support MIMO 802.11n
Atheros AR9220	-N Option	9	BPSK	24	251	-95	4800-6100	802.11a/n	Unlicensed ISM, support MIMO 802.11n
Atheros AR9220	-N Option	6	BPSK	25	316	-97	4800-6100	802.11a/n	Unlicensed ISM, support MIMO 802.11n

- M Option (SISO Multiband)

- Chipset: Atheros AR5414
- Compliant: FCC Part90Y Part 15-247/UL/ETSI 300/328/CE
- Media Access Protocol: CSMA/CA
- Modulation Technology: Orthogonal Frequency Division Multiplexing(OFDM)
  - DBPSK @ 1Mbps
  - DQPSK @ 2Mbps
  - CCK @ 5.5 & 11Mbps
  - BPSK @ 6&9Mbps
  - QPSK @ 12&18Mbps
  - 16-QAM @ 24&36Mbps
  - 64-QAM @ 48&54Mbps
- Frequency Band:2.192-2.539 GHz and 4.800-6.100 GHz
- Radio Type: Direct Sequence Spread Spectrum (DSSS)
- Current Consumption:
  - Continue Tx: </= 800mA
  - Continue Rx: </= 400mA
  - LINK Mode: </= 400mA
- Form Factor: Mini-PCI Type IIIA
- Dimensions: 59.60mm x 44.45mm
- Antenna Connector: 2x U.FL Connectors
- Operating Voltage: 5V +/- 0.15V (Input: 3.3V)
- Operating Temp: -20ºC to 70ºC
- Storage Temp: -70ºC to 85ºC
- Operating Humidity: 5%~95% Typical

Radio	Devices	Speed Mbps	Modulation	Transmit Output Power dBm	Transmit Output Power mW	Receive Sensitivity dBm	Frequency MHz	IEEE Specs	Notes
Atheros AR5414	-M Option	54	64-QAM	20	100	-70	2192-2539	802.11b/g	Unlicensed ISM, supports lower power 5GHz
Atheros AR5414	-M Option	48	64-QAM	21	126	-74	2192-2539	802.11b/g	Unlicensed ISM, supports lower power 5GHz
Atheros AR5414	-M Option	36	16-QAM	22	158	-80	2192-2539	802.11b/g	Unlicensed ISM, supports lower power 5GHz
Atheros AR5414	-M Option	24	16-QAM	24	251	-81	2192-2539	802.11b/g	Unlicensed ISM, supports lower power 5GHz
Atheros AR5414	-M Option	18	QPSK	25	316	-86	2192-2539	802.11b/g	Unlicensed ISM, supports lower power 5GHz
Atheros AR5414	-M Option	12	QPSK	25	316	-87	2192-2539	802.11b/g	Unlicensed ISM, supports lower power 5GHz
Atheros AR5414	-M Option	11	CCK	25	316	-88	2192-2539	802.11b/g	Unlicensed ISM, supports lower power 5GHz
Atheros AR5414	-M Option	9	BPSK	25	316	-89	2192-2539	802.11b/g	Unlicensed ISM, supports lower power 5GHz
Atheros AR5414	-M Option	6	BPSK	25	316	-90	2192-2539	802.11b/g	Unlicensed ISM, supports lower power 5GHz
Atheros AR5414	-M Option	5.5	CCK	25	316	-90	2192-2539	802.11b/g	Unlicensed ISM, supports lower power 5GHz
Atheros AR5414	-M Option	2	DQPSK	25	316	-91	2192-2539	802.11b/g	Unlicensed ISM, supports lower power 5GHz
Atheros AR5414	-M Option	1	DQPSK	25	316	-92	2192-2539	802.11b/g	Unlicensed ISM, supports lower power 5GHz
Atheros AR5414	-M Option	54	64-QAM	19	79	-70	4920-6100	802.11a	Unlicensed ISM, supports lower power 5GHz
Atheros AR5414	-M Option	48	64-QAM	20	100	-72	4920-6100	802.11a	Unlicensed ISM, supports lower power 5GHz
Atheros AR5414	-M Option	36	16-QAM	21	126	-79	4920-6100	802.11a	Unlicensed ISM, supports lower power 5GHz
Atheros AR5414	-M Option	24	16-QAM	21	126	-80	4920-6100	802.11a	Unlicensed ISM, supports lower power 5GHz
Atheros AR5414	-M Option	18	QPSK	24	251	-87	4920-6100	802.11a	Unlicensed ISM, supports lower power 5GHz
Atheros AR5414	-M Option	12	QPSK	24	251	-88	4920-6100	802.11a	Unlicensed ISM, supports lower power 5GHz
Atheros AR5414	-M Option	9	BPSK	24	251	-89	4920-6100	802.11a	Unlicensed ISM, supports lower power 5GHz
Atheros AR5414	-M Option	6	BPSK	24	251	-90	4920-6100	802.11a	Unlicensed ISM, supports lower power 5GHz

HauteRouterOS Basic Setup Guide

Document revision: 1.1 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

HauteRouterOS™ is independent Linux-based Operating System for wireless routers. It does not require any additional components and has no software prerequirements. It is designed with easy-to-use yet powerful interface allowing network administrators to deploy network structures and functions, that would require long education elsewhere simply by following the Reference Manual (and even without it).

Description

HauteRouterOS™ is a powerful network router software stack for use on the HR-WRAP platform. Remote control with easy real-time Windows application (HSConfig)

- Advanced Quality of Service control with burst support
- Stateful firewall with P2P protocol filtering, tunnels and IPsec
- STP bridging with filtering capabilities
- WDS and Virtual AP features
- HotSpot for Plug-and-Play access
- RIP, OSPF, BGP routing protocols
- Gigabit Ethernet ready
- V.35, X.21, T1/E1 synchronous support
- async PPP with RADUIS AAA
- IP Telephony
- remote HSConfig GUI admin
- telnet/ssh/serial console admin
- real-time configuration and monitoring
- and much more (please see the Specifications Sheet)

The Guide describes the basic steps of configuring a HR-WRAPDX running HauteRouterOS™.

Logging into the HauteSpot Router

Description

Normally you connect to the router by IP addresses with any telnet or SSH client software (a simple text-mode telnet client is usually called telnet and is distributed together with almost any OS). You can also use graphical configuration tool for Windows (also can be run in Linux using Wine) called HSConfig. To get HSConfig, connect to the router's IP address with a web browser, and follow the link to download HSConfig.exe from the router.

MAC-telnet is used to connect to a router when there is no other way to connect to it remotely if the router has no IP address or in case of misconfigured firewall. MAC-telnet can only be used from the same broadcast domain (so there should be no routers in between) as any of the router's enabled interfaces (you can not connect to a disabled interface). MAC-telnet program is a part of the Neighbor Viewer. Download it from www.hautespot.net, unpack both files contained in the archive to the same directory, and run NeighborViewer.exe. A list of HauteSpot routers working in the same broadcast domain will be showed double-click the one you need to connect to. Note that HSConfig is also able to connect to routers by their MAC addresses, and has the discovery tool built-in.

You can also connect to the router using a standard DB9 serial null-modem cable from any PC. Default settings of the router's serial port are 9600 bits/s, 8 data bits, 1 stop bit, no parity, hardware (RTS/CTS) flow control. Use terminal emulation program (like HyperTerminal or SecureCRT in Windows, or minicom in UNIX/Linux) to connect to the router. The router will beep twice when booted up, and you should see the login prompt shortly before that (check cabling and serial port settings if you do not see anything in the terminal window).

When logging into the router via terminal console, you will be presented with the HauteRouterOS™ login prompt. Use 'admin' and no password (hit [Enter]) for logging in the router for the first time, for example:

```
HauteSpot v2.9
Login: admin
Password:
```

The password can be changed with the **/password** command.

```
[admin@HauteSpot] > password
old password:
new password: ************
retype new password: ************
[admin@HauteSpot] >
```

Adding Software Packages

Description

The basic installation comes only with the **system** package. This includes basic IP routing and router administration. To have additional features such as IP Telephony, OSPF, wireless and so on, you will need to download additional software packages.

The additional software packages should have **the same version** as the system package. If not, the package won't be installed. Please consult the HauteRouterOS™ Software Package Installation and Upgrading Manual for more detailed information about installing additional software packages.

To upgrade the router packages, simply upload the packages to the router via ftp, using the binary transfer mode. After you have uploaded the packages, reboot the router, and the features that are provided by those packages will be available (regarding your license type, of course).

Navigating The Terminal Console

Description

Welcome Screen and Command Prompt

After logging into the router you will be presented with the HauteRouterOS™ Welcome Screen and command prompt, for example:

```
HauteRouterOS 2.9 (c) 1999-2004       http://www.hautespot.net/




Terminal xterm detected, using multiline input mode
[admin@HauteSpot] >
```

The command prompt shows the identity name of the router and the current menu level, for example:

```
[admin@HauteSpot] >
[admin@HauteSpot] interface>
[admin@HauteSpot] ip address>
```

Commands

The list of available commands at any menu level can be obtained by entering the question mark '?', for example:

```
[admin@HauteSpot] >
```

```
log/ -- System logs
quit -- Quit console
radius/ -- Radius client settings
certificate/ -- Certificate management
special-login/ -- Special login users
redo -- Redo previously undone action
driver/ -- Driver management
ping -- Send ICMP Echo packets
setup -- Do basic setup of system
interface/ -- Interface configuration
password -- Change password
undo -- Undo previous action
port/ -- Serial ports
import -- Run exported configuration script
snmp/ -- SNMP settings
user/ -- User management
file/ -- Local router file storage.
system/ -- System information and utilities
queue/ -- Bandwidth management
ip/ -- IP options
tool/ -- Diagnostics tools
ppp/ -- Point to Point Protocol
routing/ -- Various routing protocol settings
export  --

[admin@HauteSpot] >
[admin@HauteSpot] ip>


.. -- go up to root
service/ -- IP services
socks/ -- SOCKS version 4 proxy
arp/ -- ARP entries management
upnp/ -- Universal Plug and Play
dns/ -- DNS settings
address/ -- Address management
accounting/ -- Traffic accounting
the-proxy/ --
vrrp/ -- Virtual Router Redundancy Protocol
pool/ -- IP address pools
packing/ -- Packet packing settings
neighbor/ -- Neighbors
route/ -- Route management
firewall/ -- Firewall management
dhcp-client/ -- DHCP client settings
dhcp-relay/ -- DHCP relay settings
dhcp-server/ -- DHCP server settings
hotspot/ -- HotSpot management
ipsec/ -- IP security
web-proxy/ -- HTTP proxy
export --

[admin@HauteSpot] ip>
```

The list of available commands and menus has short descriptions next to the items. You can move to the desired menu level by typing its name and hitting the [Enter] key, for example:

```
[admin@HauteSpot] >                    | Base level menu
[admin@HauteSpot] > driver             | Enter 'driver' to move to the driver
                                       | level menu
[admin@HauteSpot] driver> /            | Enter '/' to move to the base level menu
                                       | from any level
[admin@HauteSpot] > interface          | Enter 'interface' to move to the
                                       | interface level menu
[admin@HauteSpot] interface> /ip       | Enter '/ip' to move to the IP level menu
                                       | from any level
[admin@HauteSpot] ip>                  |
```

A command or an argument does not need to be completed, if it is not ambiguous. For example, instead of typing **interface** you can type just **in** or **int**. To complete a command use the [Tab] key. Note that the completion is optional, and you can just use short command and parameter names

The commands may be invoked from the menu level, where they are located, by typing its name. If the command is in a different menu level than the current one, then the command should be invoked using its full (absolute) or relative path, for example:

```
[admin@HauteSpot] ip route> print                | Prints the routing table
[admin@HauteSpot] ip route> .. address print     | Prints the IP address table
[admin@HauteSpot] ip route> /ip address print    | Prints the IP address table
```

The commands may have arguments. The arguments have their names and values. Some commands, may have a required argument that has no name.

Summary on executing the commands and navigating the menus

Command	Action
command [Enter]	Executes the command
[?]	Shows the list of all available commands
command [?]	Displays help on the command and the list of arguments
command argument [?]	Displays help on the command's argument
[Tab]	Completes the command/word. If the input is ambiguous, a second [Tab] gives possible options
/	Moves up to the base level
/command	Executes the base level command
..	Moves up one level

""	Specifies an empty string
"word1 word2"	Specifies a string of 2 words that contain a space

You can abbreviate names of levels, commands and arguments.

For the IP address configuration, instead of using the **address** and **netmask** arguments, in most cases you can specify the address together with the number of true bits in the network mask, i.e., there is no need to specify the **netmask** separately. Thus, the following two entries would be equivalent:

```
/ip address add address 10.0.0.1/24 interface ether1
/ip address add address 10.0.0.1 netmask 255.255.255.0 interface ether1
```

Notes

You must specify the size of the network mask in the address argument, even if it is the 32-bit subnet, i.e., use **10.0.0.1/32** for `address=10.0.0.1 netmask=255.255.255.255`

Basic Configuration Tasks

Description

Interface Management

Before configuring the IP addresses and routes please check the **/interface** menu to see the list of available interfaces. If you have Plug-and-Play cards installed in the router, it is most likely that the device drivers have been loaded for them automatically, and the relevant interfaces appear on the **/interface print** list, for example:

```
[admin@HauteSpot] interface> print
Flags: X - disabled, D - dynamic, R - running
 #    NAME                TYPE            RX-RATE    TX-RATE    MTU
 0  R ether1              ether           0          0          1500
 1  R ether2              ether           0          0          1500
 2 X  wavelan1            wavelan         0          0          1500
 3 X  prism1              wlan            0          0          1500
[admin@HauteSpot] interface>
```

The interfaces need to be enabled, if you want to use them for communications. Use the **/interface enable name** command to enable the interface with a given name or number, for example:

```
[admin@HauteSpot] interface> print
Flags: X - disabled, D - dynamic, R - running
 #    NAME                        TYPE             RX-RATE    TX-RATE    MTU
 0 X  ether1                      ether            0          0          1500
 1 X  ether2                      ether            0          0          1500
[admin@HauteSpot] interface> enable 0
[admin@HauteSpot] interface> enable ether2
[admin@HauteSpot] interface> print
Flags: X - disabled, D - dynamic, R - running
 #    NAME                        TYPE             RX-RATE    TX-RATE    MTU
 0  R ether1                      ether            0          0          1500
 1  R ether2                      ether            0          0          1500
[admin@HauteSpot] interface>
```

The interface name can be changed to a more descriptive one by using **/interface set** command:

```
[admin@HauteSpot] interface> set 0 name=Local; set 1 name=Public
[admin@HauteSpot] interface> print
Flags: X - disabled, D - dynamic, R - running
 #    NAME                        TYPE             RX-RATE    TX-RATE    MTU
 0  R Local                       ether            0          0          1500
 1  R Public                      ether            0          0          1500
[admin@HauteSpot] interface>
```

Setup Command

Command name: ***/setup***

Description

The initial setup of the router can be done by using the **/setup** command which offers the following configuration:

- reset all router configuration
- load interface driver
- configure ip address and gateway
- setup dhcp client
- setup dhcp server
- setup pppoe client
- setup pptp client

Configure IP address on router, using the Setup command

Execute the **/setup** command from command line:

```
[admin@HauteSpot] > setup
  Setup uses Safe Mode. It means that all changes that are made during setup
are reverted in case of error, or if [Ctrl]+[C] is used to abort setup. To
keep
changes exit setup using the [X] key.

[Safe Mode taken]
  Choose options by pressing one of the letters in the left column, before
dash. Pressing [X] will exit current menu, pressing Enter key will select the
entry that is marked by an '*'. You can abort setup at any time by pressing
[Ctrl]+[C].
Entries marked by '+' are already configured.
Entries marked by '-' cannot be used yet.
Entries marked by 'X' cannot be used without installing additional packages.
   r - reset all router configuration
 + l - load interface driver
 * a - configure ip address and gateway
   d - setup dhcp client
   s - setup dhcp server
   p - setup pppoe client
   t - setup pptp client
   x - exit menu
your choice [press Enter to configure ip address and gateway]: a
```

To configure IP address and gateway, press **a** or [Enter], if the **a** choice is marked with an asterisk symbol ('*').

```
 * a - add ip address
 - g - setup default gateway
   x - exit menu
your choice [press Enter to add ip address]: a
```

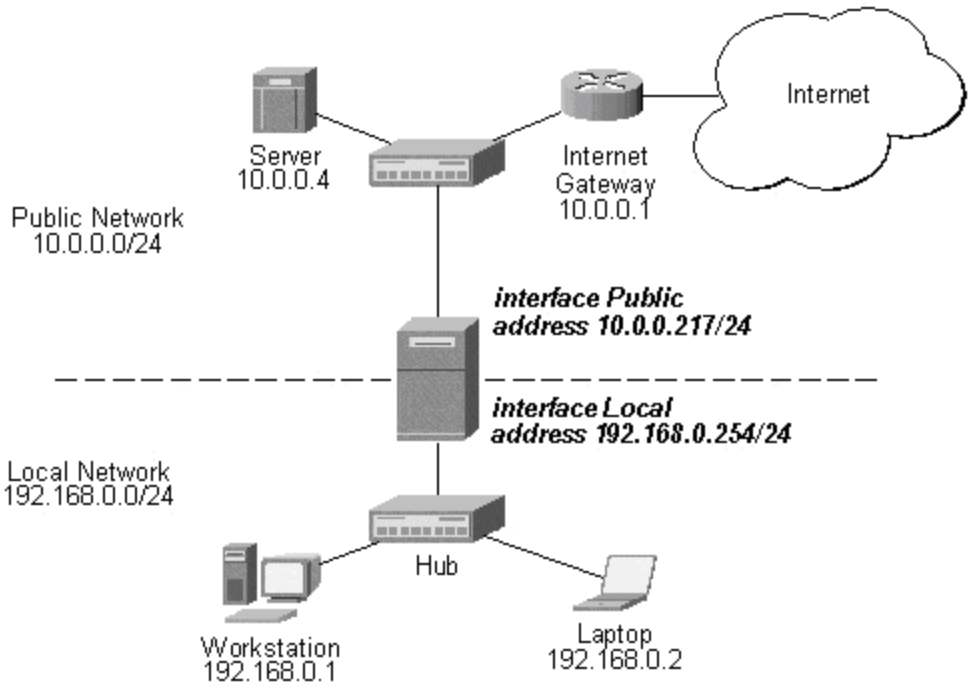
Choose **a** to add an IP address. At first, setup will ask you for an interface to which the address will be assigned. If the setup offers you an undesirable interface, erase this choice, and press the [Tab] key twice to see all available interfaces. After the interface is chosen, assign IP address and network mask on it:

```
your choice: a
enable interface:
ether1  ether2  wlan1
enable interface: ether1
ip address/netmask: 10.1.0.66/24
#Enabling interface
/interface enable ether1
#Adding IP address
/ip address add address=10.1.0.66/24 interface=ether1 comment="added by setup"
 + a - add ip address
 * g - setup default gateway
   x - exit menu
your choice: x
```

Basic Examples

Example

Assume you need to configure the HauteSpot router for the following network setup:



In the current example we use two networks:

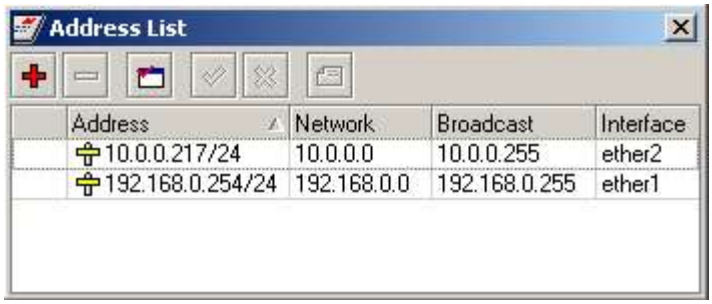
- The local LAN with network address 192.168.0.0 and 24-bit netmask: 255.255.255.0. The router's address is 192.168.0.254 in this network
- The ISP's network with address 10.0.0.0 and 24-bit netmask 255.255.255.0. The router's address is 10.0.0.217 in this network

The addresses can be added and viewed using the following commands:

```
[admin@HauteSpot] ip address> add address 10.0.0.217/24 interface Public
[admin@HauteSpot] ip address> add address 192.168.0.254/24 interface Local
[admin@HauteSpot] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
  #   ADDRESS            NETWORK         BROADCAST       INTERFACE
  0   10.0.0.217/24      10.0.0.217      10.0.0.255      Public
  1   192.168.0.254/24   192.168.0.0     192.168.0.255   Local
[admin@HauteSpot] ip address>
```

Here, the network mask has been specified in the value of the address argument. Alternatively, the argument 'netmask' could have been used with the value '255.255.255.0'. The network and broadcast addresses were not specified in the input since they could be calculated automatically.

Please **note** that the addresses assigned to different interfaces of the router should belong to different networks.



Viewing Routes

You can see two dynamic (D) and connected (C) routes, which have been added automatically when the addresses were added in the example above:

```
[admin@HauteSpot] ip route> print
Flags:  A - active, X - disabled, I - invalid, D - dynamic, C - connect,
       S - static, r - rip, b - bgp, o - ospf, d - dynamic
    #    DST-ADDRESS        G GATEWAY         DISTANCE INTERFACE
    0 ADC 192.168.0.0/24     r 0.0.0.0          0        Local
    1 ADC 10.0.0.0/24        r 0.0.0.0          0        Public
[admin@HauteSpot] ip route> print detail
Flags: A - active, X - disabled, I - invalid, D - dynamic, C - connect,
       S - static, r - rip, b - bgp, o - ospf, d - dynamic
 0 ADC dst-address=192.168.0.0/24 prefsrc=192.168.0.254 interface=Local
scope=10

 1 ADC dst-address=10.0.0.0/24 prefsrc=10.0.0.217 interface=Public scope=10

[admin@HauteSpot] ip route>
```

These routes show, that IP packets with destination to 10.0.0.0/24 would be sent through the interface Public, whereas IP packets with destination to 192.168.0.0/24 would be sent through the interface Local. However, you need to specify where the router should forward packets, which have destination other than networks connected directly to the router.

Adding Default Routes

In the following example the `default route` (destination 0.0.0.0 (any), netmask 0.0.0.0 (any)) will be added. In this case it is the ISP's gateway 10.0.0.1, which can be reached through the interface **Public**

```
[admin@HauteSpot] ip route> add gateway=10.0.0.1
[admin@HauteSpot] ip route> print
Flags: X - disabled, I - invalid, D - dynamic, J - rejected,
C - connect, S - static, R - rip, O - ospf, B - bgp
    #     DST-ADDRESS        G GATEWAY         DISTANCE INTERFACE
    0 ADC 192.168.0.0/24                               Local
    1 ADC 10.0.0.0/24                                  Public
    2 A S 0.0.0.0/0          r 10.0.0.1          0     Public
[admin@HauteSpot] ip route>
```

Here, the default route is listed under #2. As we see, the gateway 10.0.0.1 can be reached through the interface 'Public'. If the gateway was specified incorrectly, the value for the argument 'interface' would be unknown.

Notes

You cannot add two routes to the same destination, i.e., destination-address/netmask! It applies to the default routes as well. Instead, you can enter multiple gateways for one destination. For more information on IP routes, please read the Routes, Equal Cost Multipath Routing, Policy Routing manual.

If you have added an unwanted static route accidentally, use the **remove** command to delete the unneeded one. You will not be able to delete dynamic (DC) routes. They are added automatically and represent routes to the networks the router connected directly.

Testing the Network Connectivity

From now on, the **/ping** command can be used to test the network connectivity on both interfaces. You can reach any host on both connected networks from the router.

How the **/ping** command works:

```
[admin@HauteSpot] ip route> /ping 10.0.0.4
10.0.0.4 64 byte ping: ttl=255 time=7 ms
10.0.0.4 64 byte ping: ttl=255 time=5 ms
10.0.0.4 64 byte ping: ttl=255 time=5 ms
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 5/5.6/7 ms
[admin@HauteSpot] ip route>
[admin@HauteSpot] ip route> /ping 192.168.0.1
192.168.0.1 64 byte ping: ttl=255 time=1 ms
192.168.0.1 64 byte ping: ttl=255 time=1 ms
192.168.0.1 64 byte ping: ttl=255 time=1 ms
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 1/1.0/1 ms
```

```
[admin@HauteSpot] ip route>
```

The workstation and the laptop can reach (ping) the router at its local address 192.168.0.254, If the router's address 192.168.0.254 is specified as the default gateway in the TCP/IP configuration of both the workstation and the laptop, then you should be able to ping the router:

```
C:\>ping 192.168.0.254
Reply from 192.168.0.254: bytes=32 time=10ms TTL=253
Reply from 192.168.0.254: bytes=32 time<10ms TTL=253
Reply from 192.168.0.254: bytes=32 time<10ms TTL=253

C:\>ping 10.0.0.217
Reply from 10.0.0.217: bytes=32 time=10ms TTL=253
Reply from 10.0.0.217: bytes=32 time<10ms TTL=253
Reply from 10.0.0.217: bytes=32 time<10ms TTL=253

C:\>ping 10.0.0.4
Request timed out.
Request timed out.
Request timed out.
```

Notes

You cannot access anything beyond the router (network 10.0.0.0/24 and the Internet), unless you do the one of the following:

- Use source network address translation (masquerading) on the HauteSpot router to 'hide' your private LAN 192.168.0.0/24 (see the information below), or
- Add a static route on the ISP's gateway 10.0.0.1, which specifies the host 10.0.0.217 as the gateway to network 192.168.0.0/24. Then all hosts on the ISP's network, including the server, will be able to communicate with the hosts on the LAN

To set up routing, it is required that you have some knowledge of configuring TCP/IP networks. We strongly recommend that you obtain more knowledge, if you have difficulties configuring your network setups.

Advanced Configuration Tasks

Description

Next will be discussed situation with 'hiding' the private LAN 192.168.0.0/24 'behind' one address 10.0.0.217 given to you by the ISP.

Application Example with Masquerading

If you want to 'hide' the private LAN 192.168.0.0/24 'behind' one address 10.0.0.217 given to you by the ISP, you should use the source network address translation (masquerading) feature of the HauteSpot router. Masquerading is useful, if you want to access the ISP's network and the Internet appearing as all requests coming from the host 10.0.0.217 of the ISP's network. The masquerading will change the source IP address and port of the packets originated from the network 192.168.0.0/24 to the address 10.0.0.217 of the router when the packet is routed through it.

Masquerading conserves the number of global IP addresses required and it lets the whole network use a single IP address in its communication with the world.

To use masquerading, a source NAT rule with action 'masquerade' should be added to the firewall configuration:

```
[admin@HauteSpot] ip firewall nat> add chain=srcnat action=masquerade out-
interface=Public
[admin@HauteSpot] ip firewall nat> print
Flags: X - disabled, I - invalid, D - dynamic
 0   chain=srcnat out-interface=Public action=masquerade
```

Notes

Please consult Network Address Translation for more information on masquerading.

Example with Bandwidth Management

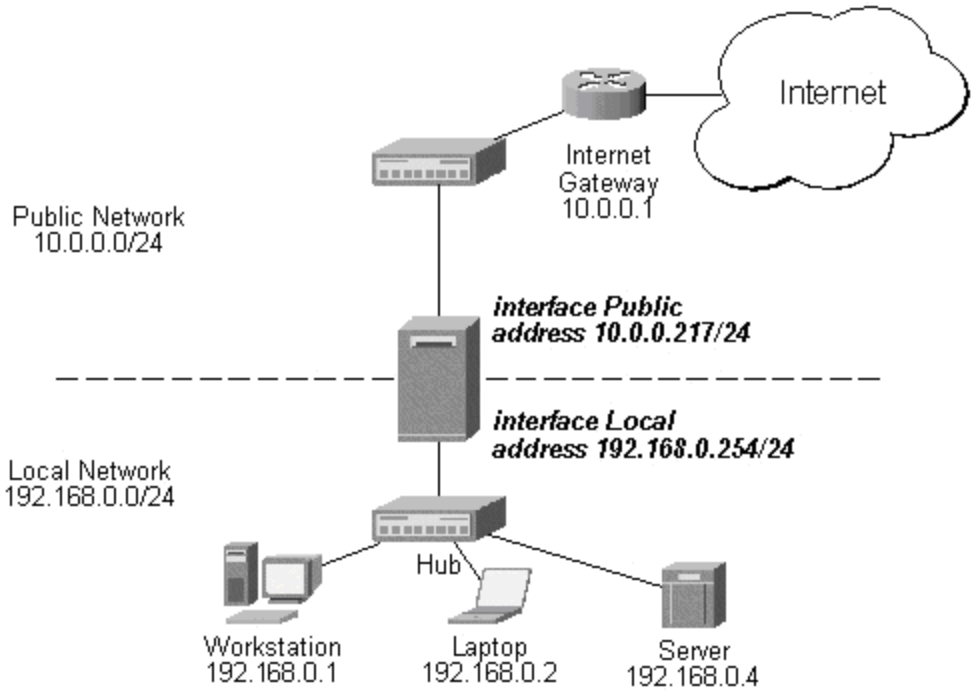
Assume you want to limit the bandwidth to 128kbps on downloads and 64kbps on uploads for all hosts on the LAN. Bandwidth limitation is done by applying queues for outgoing interfaces regarding the traffic flow. It is enough to add a single queue at the HauteSpot router:

```
[admin@HauteSpot] queue simple> add max-limit=64000/128000 interface=Local
[admin@HauteSpot] queue simple> print
Flags: X - disabled, I - invalid, D - dynamic
 0    name="queue1" target-address=0.0.0.0/0 dst-address=0.0.0.0/0
      interface=Local queue=default/default priority=8 limit-at=0/0
      max-limit=64000/128000 total-queue=default
[admin@HauteSpot] queue simple>
```

Leave all other parameters as set by default. The limit is approximately 128kbps going to the LAN (download) and 64kbps leaving the client's LAN (upload).

Example with NAT

Assume we have moved the server in our previous examples from the public network to our local one:



The server's address is now 192.168.0.4, and we are running web server on it that listens to the TCP port 80. We want to make it accessible from the Internet at address:port 10.0.0.217:80. This can be done by means of Static Network Address translation (NAT) at the HauteSpot Router. The Public address:port 10.0.0.217:80 will be translated to the Local address:port 192.168.0.4:80. One destination NAT rule is required for translating the destination address and port:

```
[admin@HauteSpot] ip firewall nat> add chain=dstnat action=dst-nat
protocol=tcp dst-address=10.0.0.217/32
                                dst-port=80 to-addresses=192.168.0.4
[admin@HauteSpot] ip firewall nat> pr
Flags: X - disabled, I - invalid, D - dynamic
 0   chain=dstnat dst-address=10.0.0.217/32 protocol=tcp dst-port=80
     action=dst-nat to-addresses=192.168.0.4 to-ports=0-65535
```

802.1q Trunk extension over Wireless P2P Link

Introduction

It is very common practice to isolate group of users in a network using VLANs. In wired

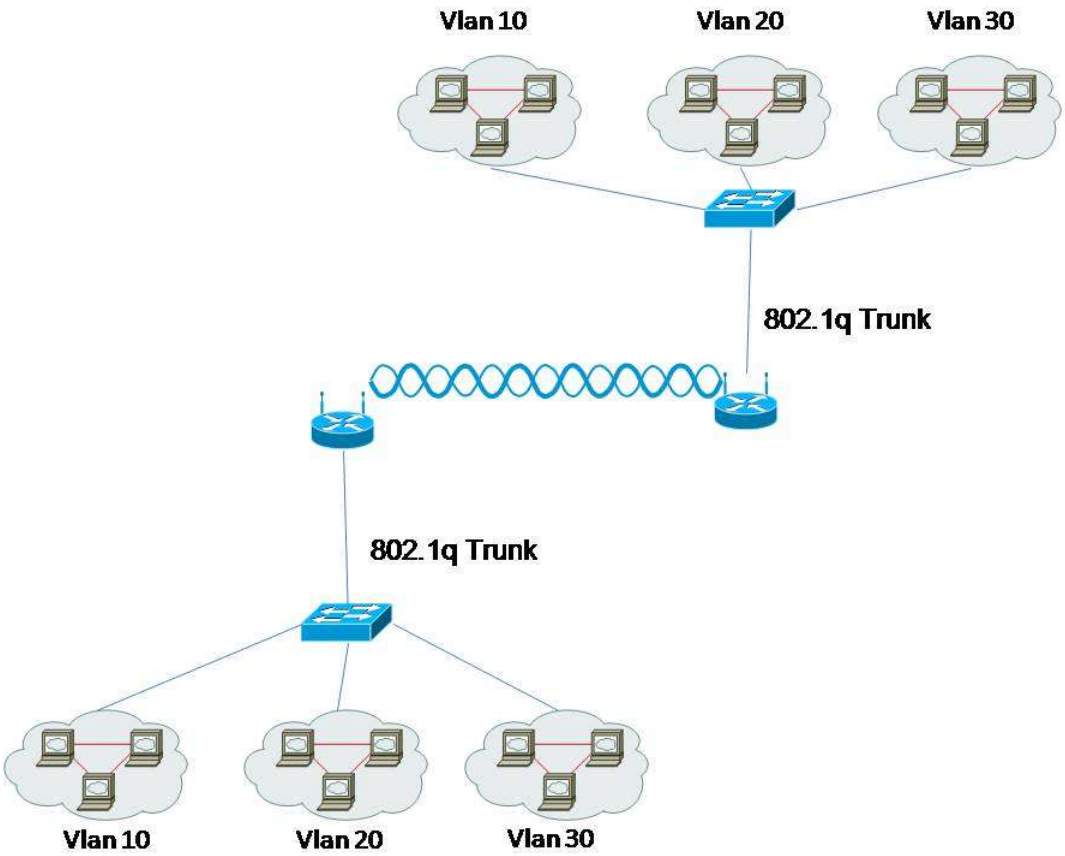
Networks we use lot of Trunk links to carry and extend VLANs from one switch to the other.

Sometimes it becomes necessary to plan redundant parallel network links for these trunk

links. Now building a trunk link using wi-fi network is little tricky.

Let us assume the Scenario as following.

- A network is having three VLANs with IDs -10,20,30
- A HauteSpot P2P link is being configured as trunk for carrying these three VLANs.
- Eth1 of the wireless routers are connected to the switches.



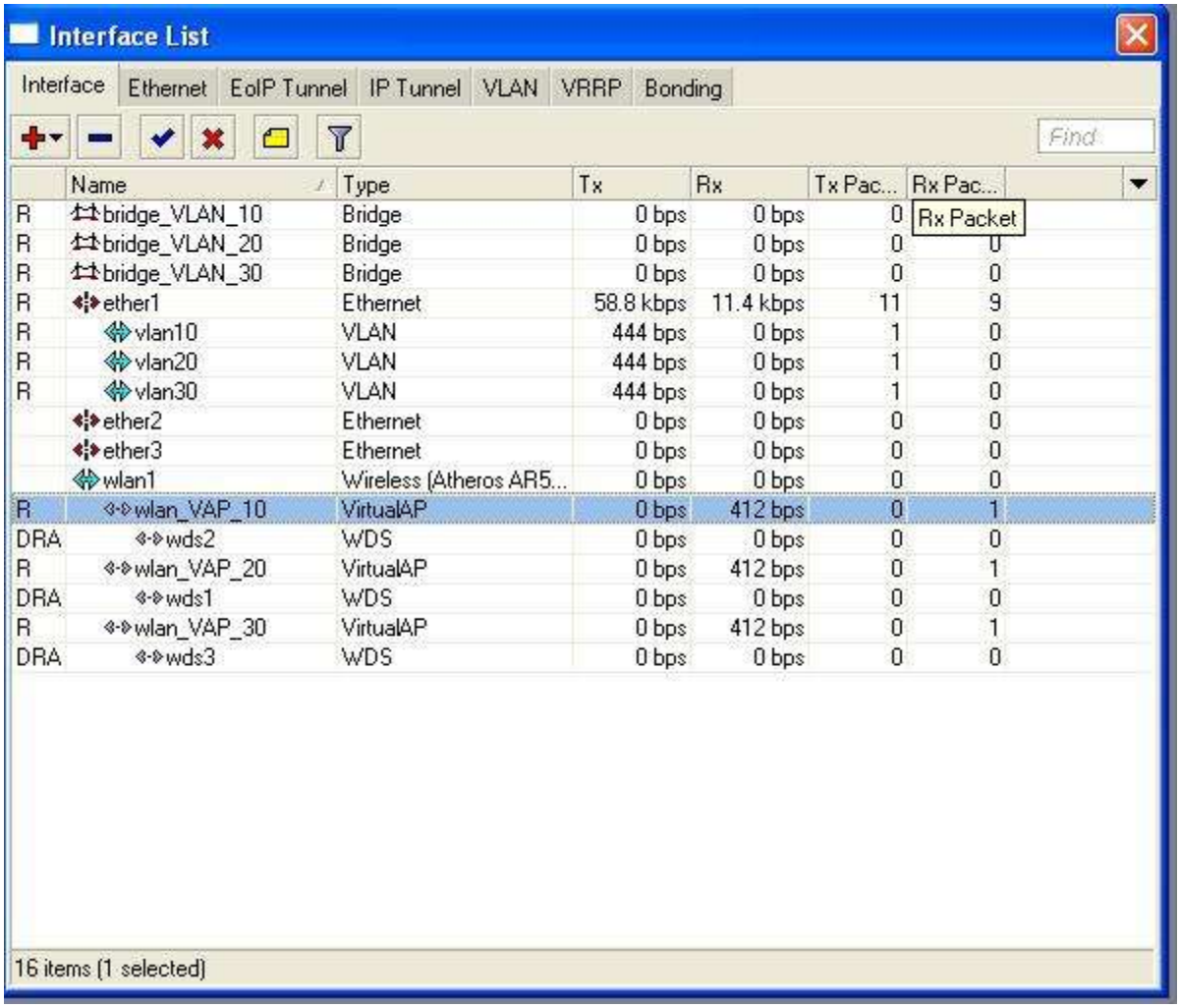
A. Create VLAN interfaces for your vlans

You need to create as many VLAN interfaces as the total no. of VLANs being forwarded through

the trunk link. Here in this example I have maintained symmetry between VLAN ID and the name

of the corresponding VLAN interface.

```
/interface vlan
add arp=enabled comment="" disabled=no interface=ether1 mtu=1500
name=vlan10 \
    vlan-id=10
add arp=enabled comment="" disabled=no interface=ether1 mtu=1500
name=vlan20 \
    vlan-id=20
add arp=enabled comment="" disabled=no interface=ether1 mtu=1500
name=vlan30 \
    vlan-id=30
```

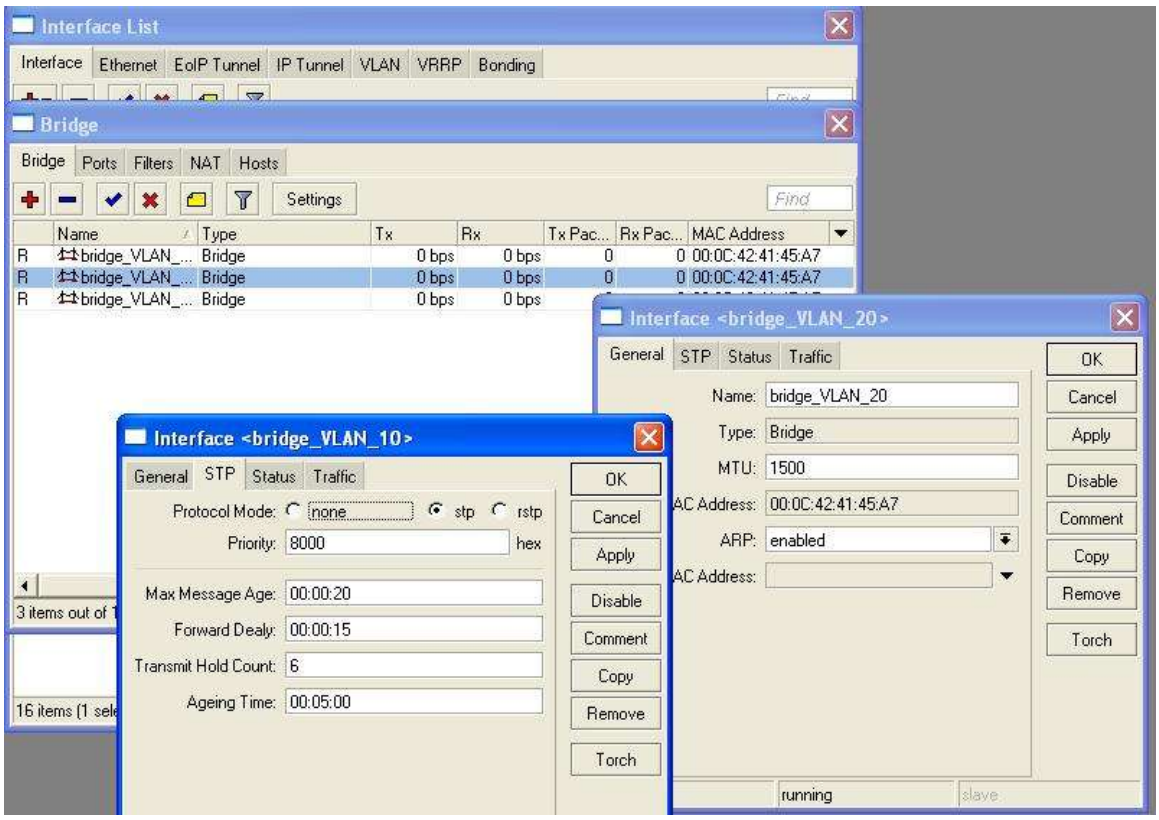

Interface List
Interface
Ethernet
EoIP Tunnel
IP Tunnel
VLAN
VRRP
Bonding
Find
Name
Type
Tx
Rx
Tx Pac...
Rx Pac...
Rx Packet
R bridge_VLAN_10 Bridge 0 bps 0 bps 0 0
R bridge_VLAN_20 Bridge 0 bps 0 bps 0 0
R bridge_VLAN_30 Bridge 0 bps 0 bps 0 0
R ether1 Ethernet 58.8 kbps 11.4 kbps 11 9
R vlan10 VLAN 444 bps 0 bps 1 0
R vlan20 VLAN 444 bps 0 bps 1 0
R vlan30 VLAN 444 bps 0 bps 1 0
ether2 Ethernet 0 bps 0 bps 0 0
ether3 Ethernet 0 bps 0 bps 0 0
wlan1 Wireless (Atheros AR5... 0 bps 0 bps 0 0
R wlan_VAP_10 VirtualAP 0 bps 412 bps 0 1
DRA wds2 WDS 0 bps 0 bps 0 0
R wlan_VAP_20 VirtualAP 0 bps 412 bps 0 1
DRA wds1 WDS 0 bps 0 bps 0 0
R wlan_VAP_30 VirtualAP 0 bps 412 bps 0 1
DRA wds3 WDS 0 bps 0 bps 0 0
16 items (1 selected)

B. Create bridges - one per vlan

One has to create as many bridges as the number of the vlan interfaces. This will run isolated Spanning Tree instances for the individual VLANs.

```
/interface bridge
add admin-mac=00:00:00:00:00:00 ageing-time=5m arp=enabled auto-mac=yes
\
    comment="" disabled=no forward-delay=15s max-message-age=20s
mtu=1500 \
    name=bridge_VLAN_10 priority=0x8000 protocol-mode=stp \
    transmit-hold-count=6
```

```
add admin-mac=00:00:00:00:00:00 ageing-time=5m arp=enabled auto-mac=yes
\
    comment="" disabled=no forward-delay=15s max-message-age=20s
mtu=1500 \
    name=bridge_VLAN_20 priority=0x8000 protocol-mode=stp \
    transmit-hold-count=6
add admin-mac=00:00:00:00:00:00 ageing-time=5m arp=enabled auto-mac=yes
\
    comment="" disabled=no forward-delay=15s max-message-age=20s
mtu=1500 \
    name=bridge_VLAN_30 priority=0x8000 protocol-mode=stp \
    transmit-hold-count=6
```

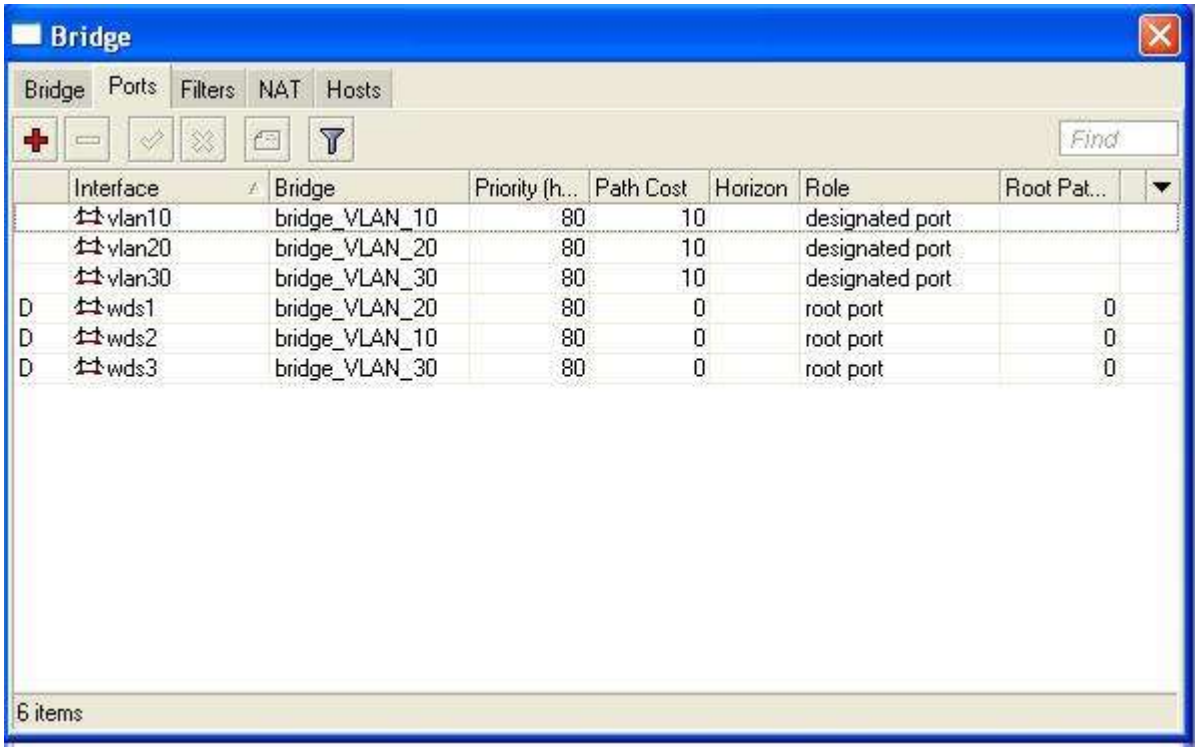


C. Add vlan interfaces to the corresponding bridge

After the creation of the bridges, vlan interfaces need to be assigned to the corresponding

bridges. This is to note that no primary interface like ether1 or wlan1 is assigned to the bridges. If this is done by mistake, the whole system might stop forwarding frames.

```
/interface bridge port
add bridge=bridge_VLAN_10 comment="" disabled=no edge=auto external-
fdb=auto \
    horizon=none interface=vlan10 path-cost=10 point-to-point=auto
priority=\
    0x80
add bridge=bridge_VLAN_20 comment="" disabled=no edge=auto external-
fdb=auto \
    horizon=none interface=vlan20 path-cost=10 point-to-point=auto
priority=\
    0x80
add bridge=bridge_VLAN_30 comment="" disabled=no edge=auto external-
fdb=auto \
    horizon=none interface=vlan30 path-cost=10 point-to-point=auto
priority=\
    0x80
```



D. Create Virtual APs - one per vlan

- The Primary interface, wlan1 in this case will be in ap-bride mode and will not have any

SSID associated with it.

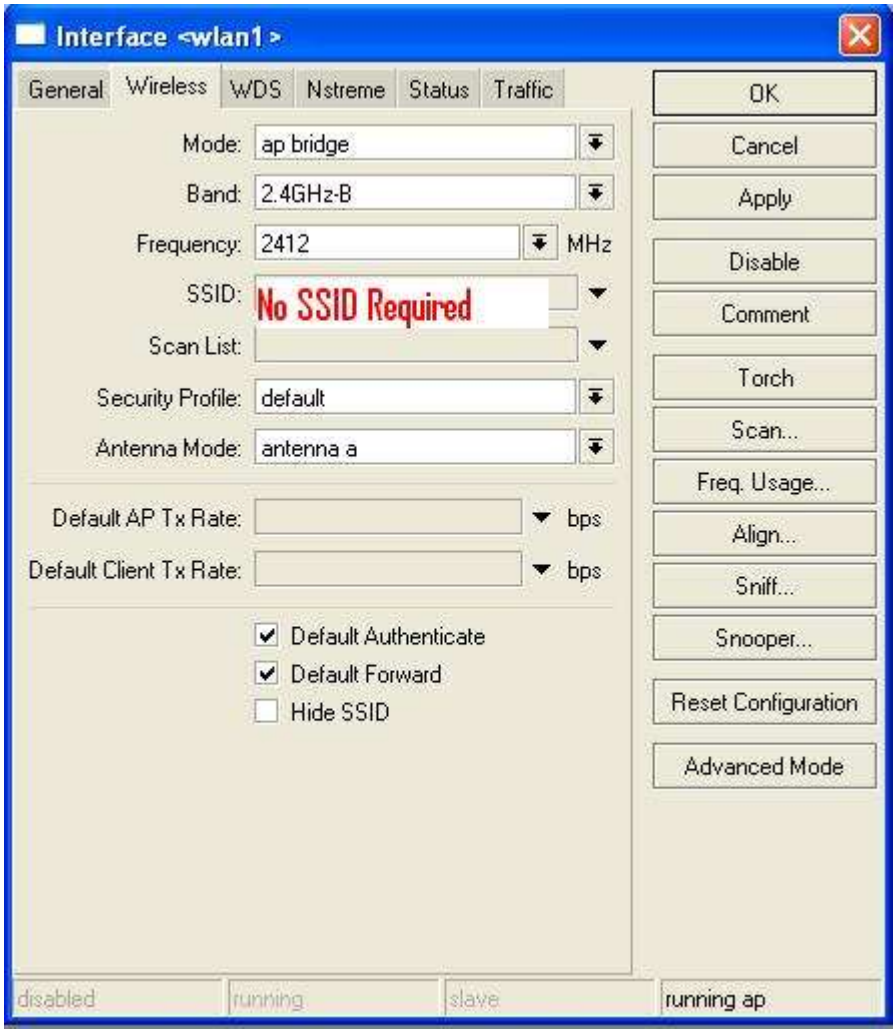
- The virtual APs will have hidden SSIDs. While creating virtual APs put 00:00:00:00:00:00

as MAC address. HauteRouterOS will dynamically assign a MAC address to these.

- In VAPs enable WDS in dynamic mode and select the corresponding VLAN bridge created

earlier.

```
/interface wireless
set 0 ack-timeout=dynamic adaptive-noise-immunity=none allow-
sharedkey=no \
    antenna-gain=0 antenna-mode=ant-a area="" arp=enabled band=2.4ghz-b
\
    basic-rates-a/g=6Mbps basic-rates-b=1Mbps burst-time=disabled
comment="" \
    compression=no country=no_country_set default-ap-tx-limit=0 \
    default-authentication=yes default-client-tx-limit=0 default-
forwarding=\
    yes dfs-mode=none disable-running-check=no disabled=no \
    disconnect-timeout=3s frame-lifetime=0 frequency=2412 frequency-
mode=\
    manual-txpower hide-ssid=no hw-retries=4 mac-
address=00:80:48:60:6B:91 \
    max-station-count=2007 mode=ap-bridge mtu=1500 name=wlan1 \
    noise-floor-threshold=default on-fail-retry-time=100ms \
    periodic-calibration=default periodic-calibration-interval=60 \
    preamble-mode=both proprietary-extensions=post-2.9.25 radio-name=\
    008048606B91 rate-set=default scan-list=default security-
profile=default \
    ssid="" station-bridge-clone-mac=00:00:00:00:00:00 supported-rates-
a/g=\
    6Mbps,9Mbps,12Mbps,18Mbps,24Mbps,36Mbps,48Mbps,54Mbps supported-
rates-b=\
    1Mbps,2Mbps,5.5Mbps,11Mbps tx-power-mode=default update-stats-
interval=\
    disabled wds-cost-range=50-150 wds-default-bridge=none wds-default-
cost=\
    100 wds-ignore-ssid=no wds-mode=disabled wmm-support=disabled
```

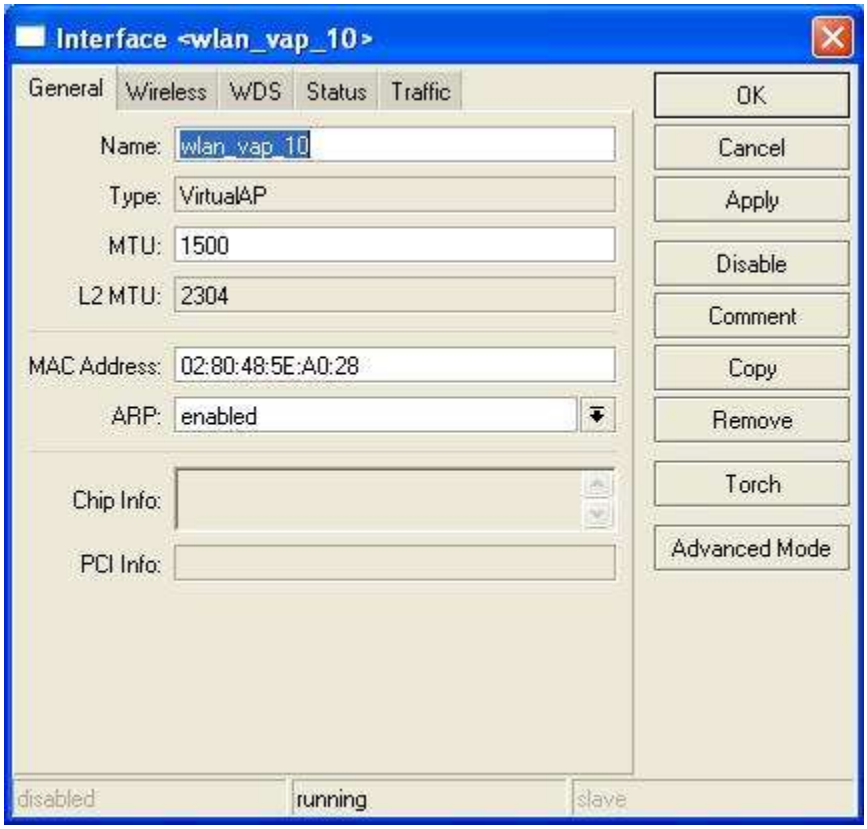


Here in this example we have created three virtual APS associated with three different bridges through three separate WDS.

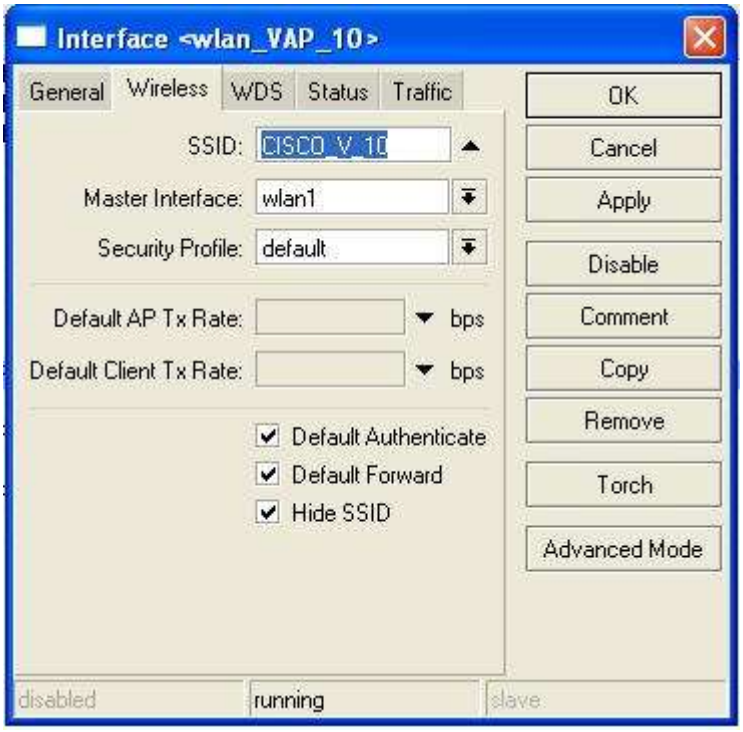
```
add area="" arp=enabled comment="" default-ap-tx-limit=0 \
    default-authentication=yes default-client-tx-limit=0 default-
forwarding=\
    yes disable-running-check=no disabled=no hide-ssid=yes mac-
address=\
    02:80:48:60:6B:91 master-interface=wlan1 max-station-count=2007
mtu=1500 \
    name=wlan_VAP_10 proprietary-extensions=post-2.9.25 security-
profile=\
    default ssid=CISCO_V_10 update-stats-interval=disabled wds-cost-
range=0 \
```

```
    wds-default-bridge=bridge_VLAN_10 wds-default-cost=0 wds-ignore-
ssid=no \
    wds-mode=dynamic wmm-support=disabled
add area="" arp=enabled comment="" default-ap-tx-limit=0 \
    default-authentication=yes default-client-tx-limit=0 default-
forwarding=\
    yes disable-running-check=no disabled=no hide-ssid=yes mac-
address=\
    02:80:48:60:6B:92 master-interface=wlan1 max-station-count=2007
mtu=1500 \
    name=wlan_VAP_20 proprietary-extensions=post-2.9.25 security-
profile=\
    default ssid=CISCO_V_20 update-stats-interval=disabled wds-cost-
range=0 \
    wds-default-bridge=bridge_VLAN_20 wds-default-cost=0 wds-ignore-
ssid=no \
    wds-mode=dynamic wmm-support=disabled
add area="" arp=enabled comment="" default-ap-tx-limit=0 \
    default-authentication=yes default-client-tx-limit=0 default-
forwarding=\
    yes disable-running-check=no disabled=no hide-ssid=yes mac-
address=\
    02:80:48:60:6B:93 master-interface=wlan1 max-station-count=2007
mtu=1500 \
    name=wlan_VAP_30 proprietary-extensions=post-2.9.25 security-
profile=\
    default ssid=CISCO_V_30 update-stats-interval=disabled wds-cost-
range=0 \
    wds-default-bridge=bridge_VLAN_30 wds-default-cost=0 wds-ignore-
ssid=no \
    wds-mode=dynamic wmm-support=disabled
```

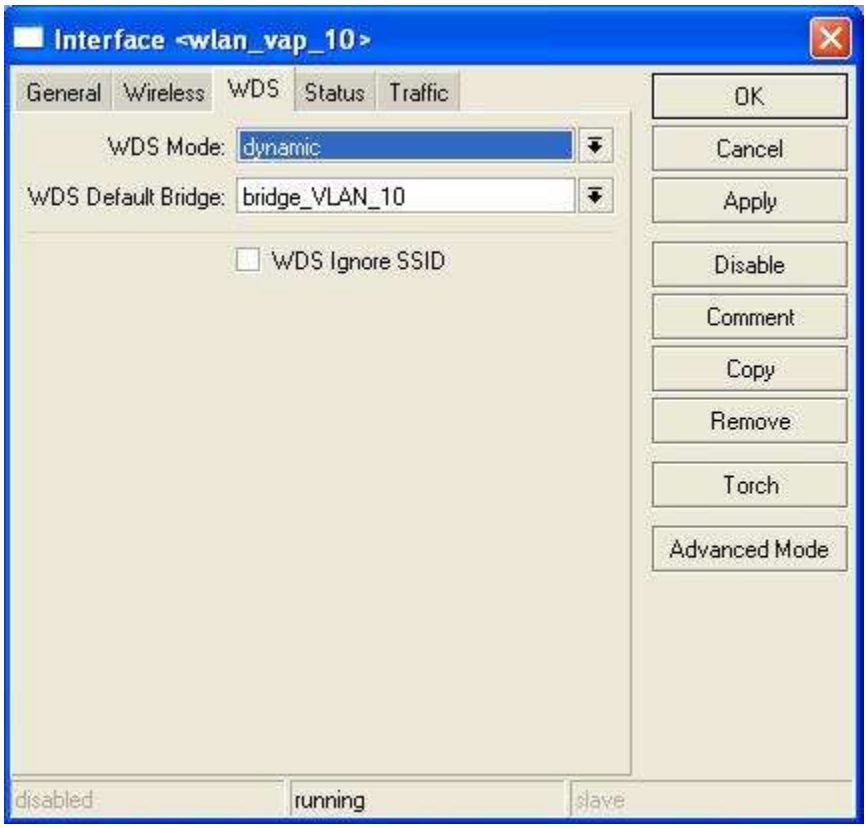
VAP General TAB: Assign a meaningful name to the Virtual AP.



VAP Wireless TAB: Create a meaningful SSID. It is advisable to hide SSID.



VAP WDS TAB: Turn on WDS in Dynamic mode and select the appropriate bridge.



Limitations

- The Aggregate Throughput of the trunk link will never be more that 22-23 Mbps. We have tested this on 5.8 GHz 802.11a only. We need to test the same on 802.11n also. We will also test TLP.

- Since one can create 128 no. of Virtual APs, total no. of VLANs that this wireless link

would be able to carry is 128 at maximum. In last 10 years I have not encountered a

situation where a trunk was carrying those many VLANs.

- This trunking will only work with 802.1q protocol. It is incompatible with ISL.

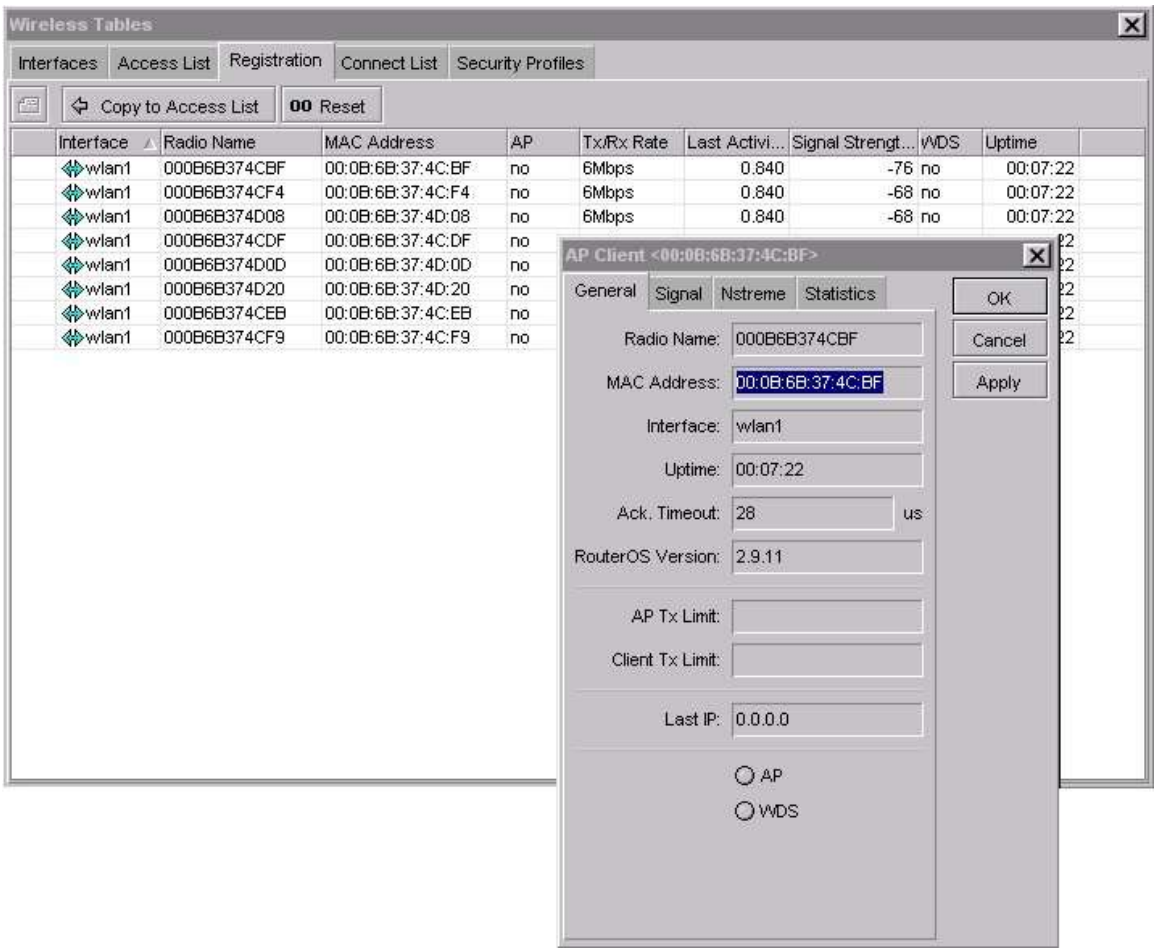
Readers' Questions and Answers

1. What about the AP configuration at the other end ? Is this article incomplete?

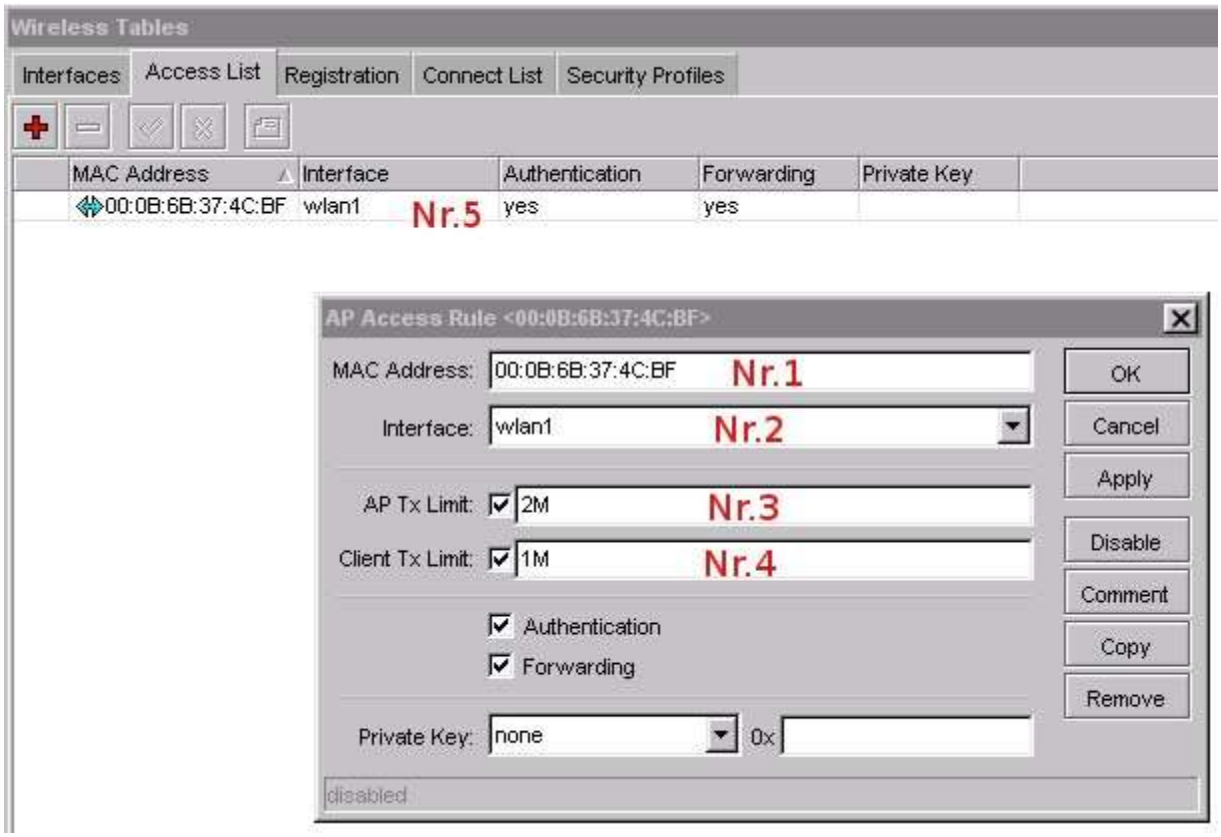
Since this is an end-to-end L-2 Set-up, both the APs would have identical configuration. Just get an export of the first AP and apply that on the second AP. Yes, it is that simple. The configuration is not incomplete. It does not have an IP address because this does not require one. Just following the steps described in this article, people were able to commission wireless trunks successfully.

Access and bandwidth limitation

Use a HauteSpot router as a wireless access point with clients associated to them. We decided allow access to wireless access point only for trusted clients and make bandwidth limit for each client. This example describew how to do it. At first look at wireless registration table and choose which client to limit.

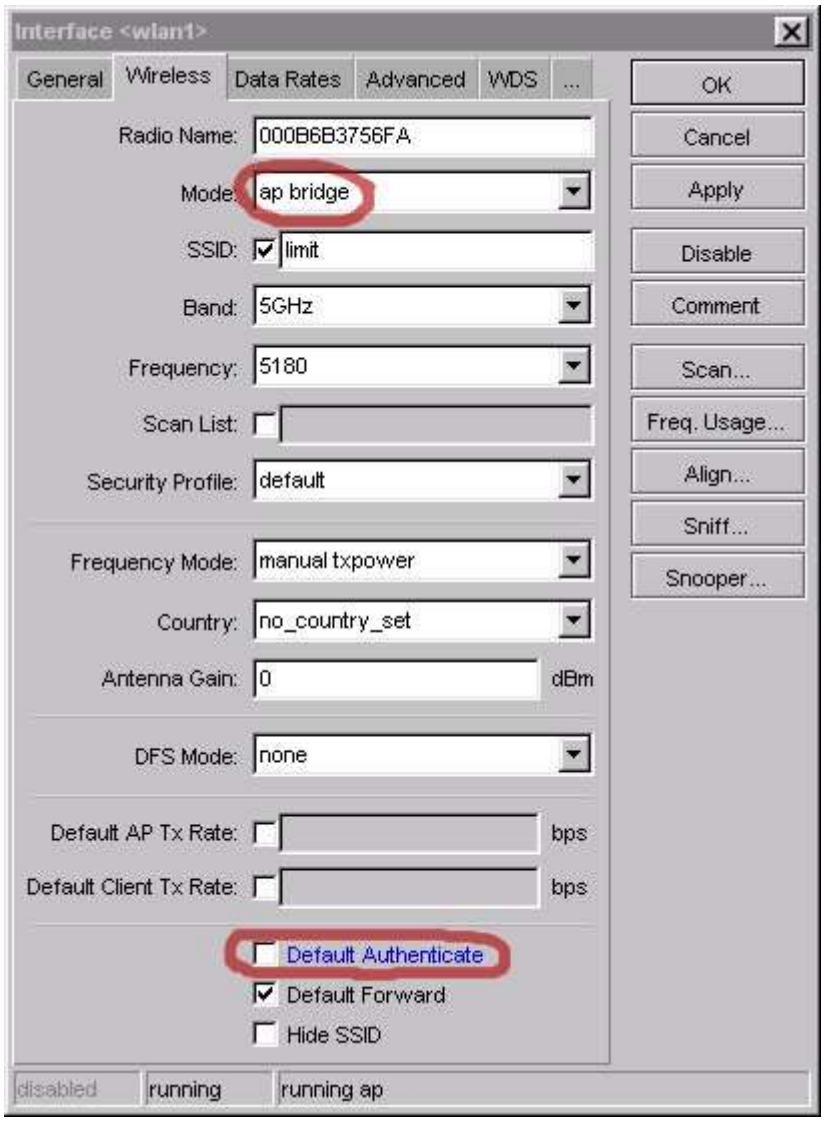


Then add this clients mac-address and limitations parameters to wireless access list. Further access list will be used for restrict associations of clients.

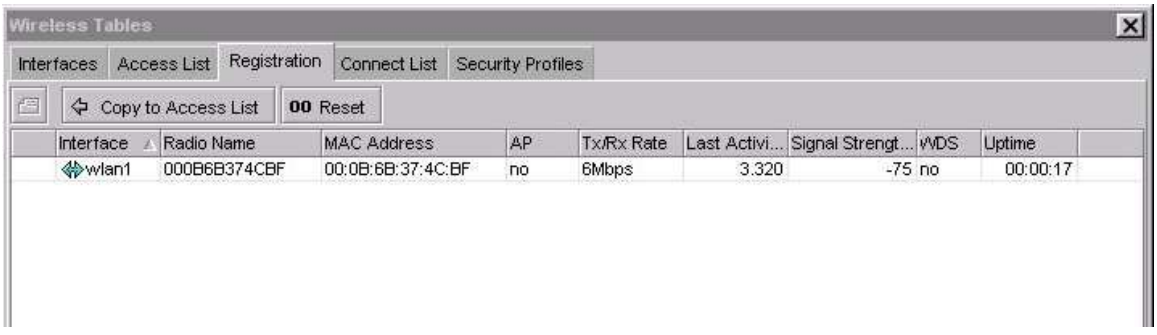



```
 Nr.1 Wireless client mac-address
 Nr.2 Wireless access point interface
 Nr.3 Limits data rate for this wireless client (0 no limit)
 Nr.4 Limits this client's transmit data rate (0 no limit) Works only
if the client is also a HauteSpot Router
 Nr.5 Wireless client is added to Access list
```

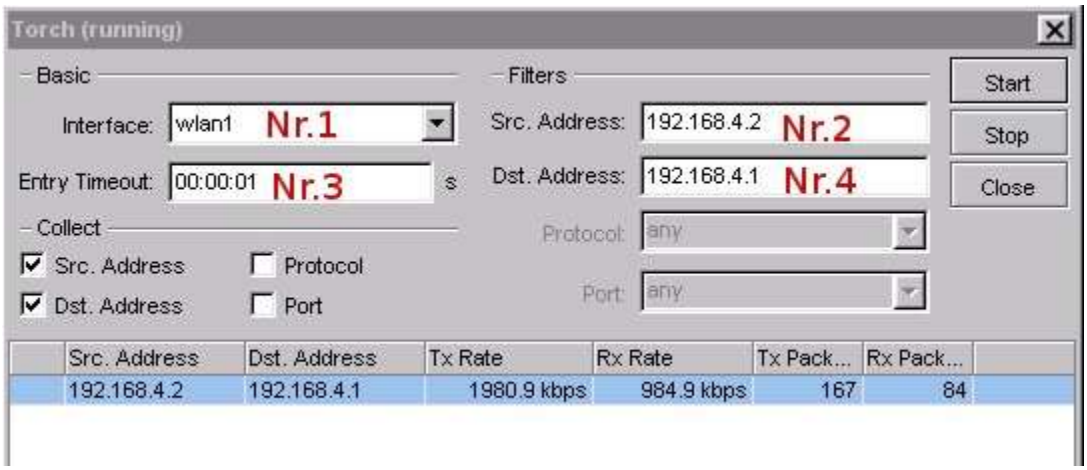
Set to wireless AP default-authentication parameter to no. It will allow authenticated clients only from access list.



Here we see wireless registration table after access list configuration.



Use torch tool to monitor bandwidth for limited wireless clients.



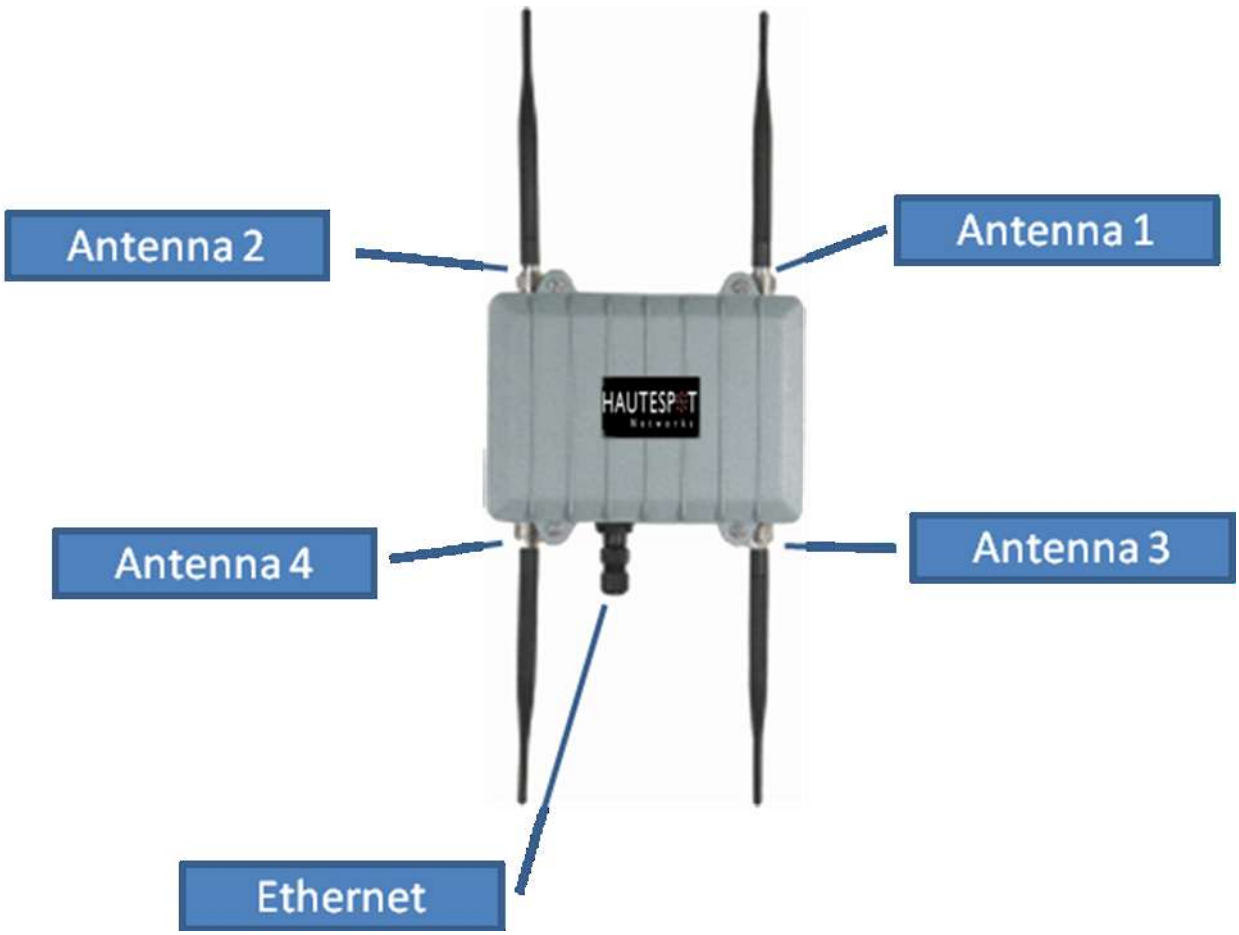

```
Nr.1 Wireless AP interface
Nr.2 Wireless client ip address
Nr.3 Time in seconds for which the screen output is paused
Nr.4 Wireless AP ip address
```

Antenna Port Utilization in HauteWRAP Family of Products

There can be anywhere from 1 to 4 antenna ports on any HauteWRAP product. These antenna ports can be assigned to radio antenna ports in a number of ways. This guide provides you with detailed information on antenna port utilization.

Outdoor System Ports

When referring to antenna ports on the outdoor versions of the HauteWRAP we will use the following nomenclature. When looking at an outdoor system from the top down with the front cover facing you and the Ethernet port at the bottom, antenna numbering starts at the upper right corner with antenna 1 and goes right to left, top to bottom, with antenna 2 in the upper left corner, antenna 3 in the lower right corner, and antenna 4 in the lower left corner.



Antenna Port Mapping to Radio Modules – Single Radio Outdoor Systems, No Diversity

When a HauteWRAP outdoor system has a single radio module installed, *without* diversity or MIMO being enabled, the main port, or port A of radio wlan1 as reported by the HauteRouterOS user interface will be connected to antenna 3

Antenna Port Mapping to Radio Modules – Single Radio Outdoor Systems with Diversity or MIMO (Special order only)

When a HauteWRAP outdoor system has a single radio module installed, *with* diversity or MIMO being enabled, the main port, or port A of radio wlan1 as reported by the HauteRouterOS user interface will be connected to antenna 1 and the auxiliary port, or port B of wlan1 as reported by the HauteRouterOS user interface will be connected to antenna 2.

Antenna Port Mapping to Radio Modules – Dual Radio Outdoor Systems, No Diversity

When a HauteWRAP outdoor system has two radio modules installed, *without* diversity or MIMO being enabled, the main port, or port A of radio wlan1 as reported by the HauteRouterOS user

interface will be connected to antenna 1 and the main port, or port A of radio wlan2 as reported by the HauteRouterOS user interface will be connected to antenna 2.

Antenna Port Mapping to Radio Modules – Dual Radio Outdoor Systems with Diversity or MIMO (Special order only)

When a HauteWRAP outdoor system has two radio modules installed, with diversity or MIMO being enabled, the main port, or port A of radio wlan1 as reported by the HauteRouterOS user interface will be connected to antenna 1 and the auxiliary port, or port B of wlan1 as reported by the HauteRouterOS user interface will be connected to antenna 2, the main port, or port A of radio wlan2 as reported by the HauteRouterOS user interface will be connected to antenna 4 and the auxiliary port, or port B of wlan2 as reported by the HauteRouterOS user interface will be connected to antenna 3.

Systems with 3G Modems

Antenna Port Mapping to Radio Modules – Single Radio Outdoor Systems, No Diversity, With 3G Modem

When a HauteWRAP outdoor system has a single radio module installed, *without* diversity or MIMO being enabled and includes a 3G modem option, the main port, or port A of radio wlan1 as reported by the HauteRouterOS user interface will be connected to antenna 1 and the 3G modem will be connected to antenna 2.

Antenna Port Mapping to Radio Modules – Single Radio Outdoor Systems with Diversity or MIMO (Special order only) With 3G Modem

When a HauteWRAP outdoor system has a single radio module installed, *with* diversity or MIMO being enabled and includes a 3G modem option, the main port, or port A of radio wlan1 as reported by the HauteRouterOS user interface will be connected to antenna 1 and the auxiliary port, or port B of wlan1 as reported by the HauteRouterOS user interface will be connected to antenna 2, and the 3G modem will be connected to antenna 4. Antenna port 3 is unused.

Antenna Port Mapping to Radio Modules – Dual Radio Outdoor Systems, No Diversity With 3G Modem

When a HauteWRAP outdoor system has two radio modules installed, *without* diversity or MIMO being enabled and includes a 3G modem option, the main port, or port A of radio wlan1 as reported by the HauteRouterOS user interface will be connected to antenna 1 and the main port, or port A of radio wlan2 as reported by the HauteRouterOS user interface will be connected to antenna 2, and the 3G modem will be connected to antenna 4. Antenna port 3 is unused.

Indoor System Ports

For indoor versions of the HauteWRAP, there are up to three antenna ports. Looking at the front panel of the system with the label up and the Ethernet facing you, the left most port is antenna 1, the center port is antenna 2, and the right port is antenna 3

Antenna Port Mapping to Radio Modules – Single Radio Indoor Systems, No Diversity

When a HauteWRAP indoor system has a single radio module installed, *without* diversity or MIMO being enabled, the main port, or port A of radio wlan1 as reported by the HauteRouterOS user interface will be connected to antenna 1.

Antenna Port Mapping to Radio Modules – Single Radio Outdoor Systems with Diversity or MIMO

When a HauteWRAP indoor system has a single radio module installed*, with* diversity or MIMO being enabled, the main port, or port A of radio wlan1 as reported by the HauteRouterOS user interface will be connected to antenna 1 and the auxiliary port, or port B of wlan1 as reported by the HauteRouterOS user interface will be connected to antenna 3.

Antenna Port Mapping to Radio Modules – Dual Radio Outdoor Systems, No Diversity

When a HauteWRAP indoor system has two radio modules installed, *without* diversity or MIMO being enabled, the main port, or port A of radio wlan1 as reported by the HauteRouterOS user interface will be connected to antenna 1 and the main port, or port A of radio wlan2 as reported by the HauteRouterOS user interface will be connected to antenna 3.

Systems with 3G Modems

Antenna Port Mapping to Radio Modules – Single Radio Outdoor Systems, No Diversity, with 3G modem option

When a HauteWRAP indoor system has one radio module installed, without diversity being enabled, the main port, or port A of radio wlan1 as reported by the HauteRouterOS user interface will be connected to antenna 1 and the 3G modem will be connected to antenna 2

Antenna Port Mapping to Radio Modules – Single Radio Outdoor Systems with Diversity or MIMO, With 3G Modem

When a HauteWRAP indoor system has a single radio module installed*, with* diversity or MIMO being enabled, the main port, or port A of radio wlan1 as reported by the HauteRouterOS user interface will be connected to antenna 1 and the auxiliary port, or port B of wlan1 as reported by the HauteRouterOS user interface will be connected to antenna 3, and the 3G modem is connected to antenna 2.

Antenna Port Mapping to Radio Modules – Dual Radio Outdoor Systems, No Diversity, With 3G Modem

When a HauteWRAP indoor system has two radio modules installed, *without* diversity or MIMO being enabled, the main port, or port A of radio wlan1 as reported by the HauteRouterOS user interface will be connected to antenna 1 and the main port, or port A of radio wlan2 as reported by the HauteRouterOS user interface will be connected to antenna 3, and the 3G modem is connected to antenna 2.

Radio Module Nomenclature

Every HauteWRAP product has a set of numbers at the end of it's part code. These numbers indicate the radio modules being using in the router. The first number indicates the radio used for wlan1 and the second number indicates the radio used for wlan2.

-M	2.4 GHz 25dBm/5GHz 24dBm dual band radio
-N	2.4 GHz 25dBm/5GHz 25dBm 2x2 MIMO dual band radio
-2	2.4GHz 28dBm only
-3	3.65GHz 25dBm only
-4	4.9GHz 25dBm only
-5	5GHz 28dBm only
-7	700MHz 28dBm only
-9	900MHz 28dBm only

Product Antenna Mapping Details

The following table shows antenna port mappings for various HauteWRAP products:

Outdoor Products

Product	Antenna Port 1	Antenna Port 2	Antenna Port 3	Antenna Port 4
HR-WRAPSXC-M HR-WRAPSXC3E-M HR-WRAPLX-M			2.4/5GHz wlan1 A	
HR-WRAPSXC-2 HR-WRAPSXC3E-2 HR-WRAPLX-2			2.4 GHz wlan1 A	
HR-WRAPSXC-37 HR-WRAPSXC3E-37 HR-WRAPLX-37			3.65GHz wlan1 A	
HR-WRAPSXC-4 HR-WRAPSXC3E-4 HR-WRAPLX-4			4.9GHz wlan1 A	
HR-WRAPSXC-5 HR-WRAPSXC3E-5 HR-WRAPLX-5			5 GHz wlan1 A	
HR-WRAPSXC-9 HR-WRAPSXC3E-9 HR-WRAPLX-9			900MHz wlan1 A	
HR-WRAPDXC-M HR-WRAPDXC3E-M	2.4/5GHz wlan1 A	2.4/5GHz wlan2 A		
HR-WRAPDXC-2M HR-WRAPDXC3E-2M	2.4 GHz wlan1 A	2.4/5GHz wlan2 A		
HR-WRAPDXC-37M HR-WRAPDXC3E-37M	3.65 GHz wlan1 A	2.4/5GHz wlan2 A		
HR-WRAPDXC-4M HR-WRAPDXC3E-4M	4.9GHz wlan2 A	2.4/5GHz wlan2 A		
HR-WRAPDXC-5M HR-WRAPDXC3E-5M	5 GHz wlan2 A	2.4/5GHz wlan2 A		
HR-WRAPDXC-9M HR-WRAPDXC3E-9M	900MHz wlan2 A	2.4/5GHz wlan2 A		
HR-WRAPDXC-22 HR-WRAPDXC3E-22	2.4 GHz wlan1 A	2.4 GHz wlan2 A		
HR-WRAPDXC-24 HR-WRAPDXC3E-24	2.4 GHz wlan1 A	4.9GHz wlan2 A		
HR-WRAPDXC-25 HR-WRAPDXC3E-25	2.4 GHz wlan1 A	5 GHz wlan2 A		
HR-WRAPDXC-29 HR-WRAPDXC3E-29	2.4 GHz wlan1 A	900MHz wlan2 A		
HR-WRAPDXC-44 HR-WRAPDXC3E-44	4.9GHz wlan1 A	4.9GHz wlan2 A		
HR-WRAPDXC-45 HR-WRAPDXC3E-45	4.9GHz wlan1 A	5 GHz wlan2 A		
HR-WRAPDXC-49 HR-WRAPDXC3E-49	4.9GHz wlan1 A	900MHz wlan2 A		
HR-WRAPDXC-55 HR-WRAPDXC3E-55	5 GHz wlan1 A	5 GHz wlan2 A		
HR-WRAPDXC-59 HR-WRAPDXC3E-59	5 GHz wlan1 A	900MHz wlan2 A		
HR-WRAPDXC-99 HR-WRAPDXC3E-99	900MHz wlan1 A	900MHz wlan2 A		

Outdoor Products with Diversity or MIMO

Product	Antenna Port 1	Antenna Port 2	Antenna Port 3	Antenna Port 4
HR-WRAPSXCs-M	2.4/5GHz wlan1 A	2.4/5GHz wlan1 B		
HR-WRAPSXCs-9	900MHz wlan1 A	900MHz wlan1 B		
HR-WRAPSXC-N HR-WRAPSXC3E-N HR-WRAPLX-N	2.4/5GHz MIMO wlan1 A	2.4/5GHz MIMO wlan1 B		
HR-WRAPDXC-N HR-WRAPDXC3E-N	2.4/5GHz MIMO wlan1 A	2.4/5GHz MIMO wlan1 B	2.4/5GHz MIMO wlan2 A	2.4/5GHz MIMO wlan2 B
HR-WRAPDXC-37N HR-WRAPDXC3E-37N	2.4/5GHz MIMO wlan1 A	2.4/5GHz MIMO wlan1 B	3.65GHz wlan2 A	
HR-WRAPDXC-4N HR-WRAPDXC3E-4N	2.4/5GHz MIMO wlan1 A	2.4/5GHz MIMO wlan1 B	4GHz wlan2 A	
HR-WRAPDXC-5N HR-WRAPDXC3E-5N	2.4/5GHz MIMO wlan1 A	2.4/5GHz MIMO wlan1 B	5GHz wlan2 A	
HR-WRAPDXC-9N HR-WRAPDXC3E-9N	2.4/5GHz MIMO wlan1 A	2.4/5GHz MIMO wlan1 B	900MHz wlan2 A	

Outdoor Products with 3G modems

Product	Antenna Port 1	Antenna Port 2	Antenna Port 3	Antenna Port 4
HR-WRAPSXC-M3G HR-WRAPSXC3E-M3G	2.4/5GHz wlan1 A	3G modem		
HR-WRAPSXC-N3G HR-WRAPSXC3E-N3G	2.4/5GHz wlan1 A	2.4/5GHz wlan1 B		3G modem
HR-WRAPSXC-373G HR-WRAPSXC3E-373G	3.65GHz wlan1 A	3G modem		
HR-WRAPSXC-93G HR-WRAPSXC3E-93G	900MHz wlan1 A	3G modem		
HR-WRAPDXC-M3G HR-WRAPDXC3E-M3G	2.4/5GHz wlan1 A	2.4/5GHz wlan2 A		3G modem
HR-WRAPDXC-37M3G HR-WRAPDXC3E-37M3G	3.65 GHz wlan1 A	3.65GHz wlan2 A		3G modem
HR-WRAPDXC-4M3G HR-WRAPDXC3E-4M3G	4.9GHz wlan2 A	2.4/5GHz wlan2 A		3G modem
HR-WRAPDXC-5M3G HR-WRAPDXC3E-5M3G	5GHz wlan2 A	2.4/5GHz wlan2 A		3G modem
HR-WRAPDXC-9M3G HR-WRAPDXC3E-9M3G	900MHz wlan2 A	2.4/5GHz wlan2 A		3G modem

Outdoor Products with MIMO and 3G support

Product	Antenna Port 1	Antenna Port 2	Antenna Port 3	Antenna Port 4
HR-WRAPSXC-N3G HR-WRAPSXC3E-N3G	2.4/5GHz MIMO wlan1 A	2.4/5GHz MIMO wlan1 B		3G modem
HR-WRAPDXC-2N3G HR-WRAPDXC3E-2N3G	2.4/5GHz MIMO wlan1 A	2.4/5GHz MIMO wlan1 B	2.4GHz wlan2 A	3G modem
HR-WRAPDXC-37N3G HR-WRAPDXC3E-37N3G	2.4/5GHz MIMO wlan1 A	2.4/5GHz MIMO wlan1 B	3.65GHz wlan2 A	3G modem
HR-WRAPDXC-4N3G HR-WRAPDXC3E-4N3G	2.4/5GHz MIMO wlan1 A	2.4/5GHz MIMO wlan1 B	4.9GHz wlan2 A	3G modem
HR-WRAPDXC-5N3G HR-WRAPDXC3E-5N3G	2.4/5GHz MIMO wlan1 A	2.4/5GHz MIMO wlan1 B	5GHz wlan2 A	3G modem
HR-WRAPDXC-9N3G HR-WRAPDXC3E-9N3G	2.4/5GHz MIMO wlan1 A	2.4/5GHz MIMO wlan1 B	900MHz wlan2 A	3G modem

Indoor Products

Product	Antenna Port 1	Antenna Port 2	Antenna Port 3
HR-WRAPSXCi-M	2.4/5GHz wlan1 A		
HR-WRAPSXCi-2	2.4 GHz wlan1 A		
HR-WRAPSXCi-37	3.65GHz wlan1 A		
HR-WRAPSXCi-4	4.9GHz wlan1 A		
HR-WRAPSXCi-5	5 GHz wlan1 A		
HR-WRAPSXCi-9	900MHz wlan1 A		
HR-WRAPDXCi-M	2.4/5GHz wlan1 A		2.4/5GHz wlan2 A
HR-WRAPDXCi-2M	2.4 GHz wlan1 A		2.4/5GHz wlan2 A
HR-WRAPDXCi-37M	3.65 GHz wlan1 A		2.4/5GHz wlan2 A
HR-WRAPDXCi-4M	4.9GHz wlan2 A		2.4/5GHz wlan2 A
HR-WRAPDXCi-5M	5 GHz wlan2 A		2.4/5GHz wlan2 A
HR-WRAPDXCi-9M	900MHz wlan2 A		2.4/5GHz wlan2 A
HR-WRAPDXCi-22	2.4 GHz wlan1 A		2.4 GHz wlan2 A
HR-WRAPDXCi-24	2.4 GHz wlan1 A		4.9GHz wlan2 A
HR-WRAPDXCi-25	2.4 GHz wlan1 A		5 GHz wlan2 A
HR-WRAPDXCi-29	2.4 GHz wlan1 A		900MHz wlan2 A
HR-WRAPDXCi-44	4.9GHz wlan1 A		4.9GHz wlan2 A
HR-WRAPDXCi-45	4.9GHz wlan1 A		5 GHz wlan2 A
HR-WRAPDXCi-49	4.9GHz wlan1 A		900MHz wlan2 A
HR-WRAPDXCi-55	5 GHz wlan1 A		5 GHz wlan2 A
HR-WRAPDXCi-59	5 GHz wlan1 A		900MHz wlan2 A
HR-WRAPDXCi-99	900MHz wlan1 A		900MHz wlan2 A

Indoor Products with MIMO

Product	Antenna Port 1	Antenna Port 2	Antenna Port 3
HR-WRAPSXCi-N	2.4/5GHz MIMO wlan1 A	2.4/5GHz MIMO wlan1 B	
HR-WRAPDXC-2N	2.4/5GHz MIMO wlan1 A	2.4/5GHz MIMO wlan1 B	2.4GHz wlan2 A
HR-WRAPDXC-37N	2.4/5GHz MIMO wlan1 A	2.4/5GHz MIMO wlan1 B	3.65GHz wlan2 A
HR-WRAPDXC-4N	2.4/5GHz MIMO wlan1 A	2.4/5GHz MIMO wlan1 B	4GHz wlan2 A
HR-WRAPDXC-5N	2.4/5GHz MIMO wlan1 A	2.4/5GHz MIMO wlan1 B	5GHz wlan2 A
HR-WRAPDXC-9N	2.4/5GHz MIMO wlan1 A	2.4/5GHz MIMO wlan1 B	900MHz wlan2 A

Indoor Products with 3G modems

Product	Antenna Port 1	Antenna Port 2	Antenna Port 3
HR-WRAPSXCi-M	2.4/5GHz wlan1 A	3G modem	
HR-WRAPSXCi-2	2.4 GHz wlan1 A	3G modem	
HR-WRAPSXCi-37	3.65GHz wlan1 A	3G modem	
HR-WRAPSXCi-4	4.9GHz wlan1 A	3G modem	
HR-WRAPSXCi-5	5 GHz wlan1 A	3G modem	
HR-WRAPSXCi-9	900MHz wlan1 A	3G modem	
HR-WRAPDXCi-M	2.4/5GHz wlan1 A	3G modem	2.4/5GHz wlan2 A
HR-WRAPDXCi-2M	2.4 GHz wlan1 A	3G modem	2.4/5GHz wlan2 A
HR-WRAPDXCi-37M	3.65 GHz wlan1 A	3G modem	2.4/5GHz wlan2 A
HR-WRAPDXCi-4M	4.9GHz wlan2 A	3G modem	2.4/5GHz wlan2 A
HR-WRAPDXCi-5M	5 GHz wlan2 A	3G modem	2.4/5GHz wlan2 A
HR-WRAPDXCi-9M	900MHz wlan2 A	3G modem	2.4/5GHz wlan2 A
HR-WRAPDXCi-22	2.4 GHz wlan1 A	3G modem	2.4 GHz wlan2 A
HR-WRAPDXCi-24	2.4 GHz wlan1 A	3G modem	4.9GHz wlan2 A
HR-WRAPDXCi-25	2.4 GHz wlan1 A	3G modem	5 GHz wlan2 A
HR-WRAPDXCi-29	2.4 GHz wlan1 A	3G modem	900MHz wlan2 A
HR-WRAPDXCi-44	4.9GHz wlan1 A	3G modem	4.9GHz wlan2 A
HR-WRAPDXCi-45	4.9GHz wlan1 A	3G modem	5 GHz wlan2 A
HR-WRAPDXCi-49	4.9GHz wlan1 A	3G modem	900MHz wlan2 A
HR-WRAPDXCi-55	5 GHz wlan1 A	3G modem	5 GHz wlan2 A
HR-WRAPDXCi-59	5 GHz wlan1 A	3G modem	900MHz wlan2 A

HR-WRAPDXCi-99	900MHz wlan1 A	3G modem	900MHz wlan2 A

Indoor Products with MIMO and 3G support

Product	Antenna Port 1	Antenna Port 2	Antenna Port 3
HR-WRAPSXCi-N3G	2.4/5GHz MIMO wlan1 A	3G modem	2.4/5GHz MIMO wlan1 B

HauteRouterOS Bandwidth Control

Document revision: 2.2 (November 28, 2007, 10:45 GMT)

Applies to: V3.0

General Information

Summary

Bandwidth Control is a set of mechanisms that control data rate allocation, delay variability, timely delivery, and delivery reliability. The HauteSpot RouterOS supports the following queuing disciplines:

- **PFIFO** - Packets First-In First-Out
- **BFIFO** - Bytes First-In First-Out
- **SFQ** - Stochastic Fairness Queuing
- **RED** - Random Early Detect
- **PCQ** - Per Connection Queue
- **HTB** - Hierarchical Token Bucket

Specifications

Packages required: ***system***
License required: *Level1 (limited to 1 queue)* , *Level3*
Submenu level: ***/queue***
Standards and Technologies: None
Hardware usage: *significant*

Description

Quality of Service (QoS) means that the router should prioritize and shape network traffic. QoS is not so much about limiting, it is more about providing quality service to the network users. Some features of HauteSpot RouterOS traffic control mechanism are listed below:

- limit data rate for certain IP addresses, subnets, protocols, ports, and other parameters
- limit peer-to-peer traffic
- prioritize some packet flows over others
- use queue bursts for faster web browsing
- apply queues on fixed time intervals
- share available traffic among users equally, or depending on the load of the channel

The queuing is applied on packets leaving the router through a real interface (i.e., the queues are applied on the outgoing interface, regarding the traffic flow), or any of the 3 additional virtual interfaces (global-in, global-out, global-total).
The QoS is performed by means of dropping packets. In case of TCP protocol, the dropped packets will be resent on a slower rate, so there is no need to worry that with shaping we lose some TCP information.
The main terms used to describe the level of QoS for network applications, are:

- **queuing discipline (qdisc)** - an algorithm that holds and maintains a queue of packets. It accumulates the packets and decides the order of the outgoing packets (it means that queuing discipline can reorder packets). Qdisc also decides which packets to drop if there is no space for them.
- **CIR (Committed Information Rate)** - the guaranteed data rate. It means that traffic rate, not exceeding this value should always be delivered
- **MIR (Maximal Information Rate)** - the maximal data rate router will provide
- **Priority** - the order of importance in what traffic will be processed. You can give priority to some traffic in order it to be handled before some other traffic
- **Contention Ratio** - the ratio to which the defined data rate is shared among users (when a certain data rate is allocated to a number of subscribers). It is the number of subscribers that have a single speed limitation, applied to all of them together. For example, the contention ratio of 1:4 means that the allocated data rate may be shared between no more than 4 users

Before sending data over an interface, it is processed with a queuing discipline. There can be only one queuing discipline per interface, which, by default, is set under **/queue interface** for each physical interface (there is no default queuing discipline for virtual interfaces). Once we add a first queue (in **/queue tree** or **/queue simple**) to a physical interface, the interface default queue is replaced by HTB hierarchy with that queue, but the one defined in **/queue interface** for that particular interface, is no more active.

Scheduler and Shaper qdiscs
We can classify queuing disciplines by their influence to packet flow:

- **schedulers** - queuing disciplines only reschedule packets regarding their algorithm and drop packets which 'do not fit in the queue'. Scheduler queuing disciplines are: PFIFO, BFIFO, SFQ, PCQ (both scheduler and shaper), RED
- **shapers** - queuing disciplines that also perform the limitation. Shapers are PCQ (both scheduler and shaper) and HTB

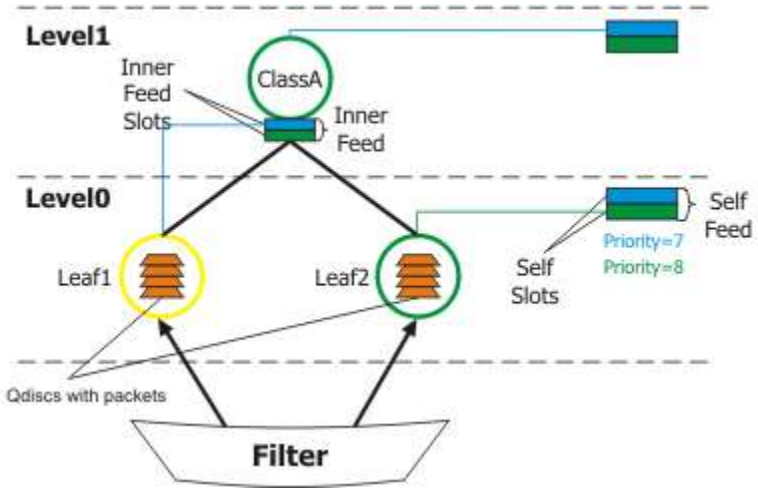
Virtual Interfaces
There are 3 virtual interfaces in RouterOS, in addition to real interfaces:

- **global-in** - represents all the input interfaces in general (INGRESS queue). Please **note** that queues attached to **global-in** apply to traffic that is received by the router, before the packet filtering. **global-in** queuing is executed just after mangle and dst-nat
- **global-out** - represents all the output interfaces in general (EGRESS queue). Queues attached to it apply before the ones attached to a specific interface
- **global-total** - represents a virtual interface through which all the data, going through the router, is passing. When attaching a qdisc to global-total, the limitation is done in both directions. For example, if we set a **total-max-limit** to 256000, we will get upload+download=256kbps (maximum)

Introduction to HTB

HTB (Hierarchical Token Bucket) is a classful queuing discipline that is useful for applying different handling for different kinds of traffic. The queues you add in **/queue simple** and **/queue tree** are attached to the main Hierarchical Token Bucket (HTB). For example, you can set a maximum data rate for a workgroup and then distribute that amount of traffic between the members of that workgroup.

HTB qdisc in detail:



HTB terms:

- **filter** - a procedure that classifies packets. The filters are responsible for classifying packets so that they are put in the corresponding qdiscs. All filters are applied at the HTB root and classify packets directly into the qdiscs, without traversing the HTB tree. If a packet is not classified into any of the qdiscs, it is sent out to the interface directly, traversing the HTB, so no HTB rules are applied to those packets (it would mean effective higher priority than of any packet flow managed by HTB).
- **level** - position of a class in the hierarchy.
- **class** - algorithm for limiting traffic flow to a certain rate. It does not store any packets (this function can only be performed by a queue). A class may contain either one or more subclasses (inner class), or one and only one qdisc (leaf class).
- **inner class** - a class that has one or more child class attached to it. As inner classes do not store any packets, qdiscs can not be attached to them (so their qdisc and filter settings are ignored, although may be still shown in RouterOS configuration), so they only do traffic shaping. Priority setting is ignored as well.
- **leaf class** - a class that has a parent but does not have any child classes. Leaf classes are always located at level 0 of the hierarchy. Each leaf class has one and only one qdisc attached to it, with a certain priority.
- **self feed** - an exit (out of the HTB tree, to the interface) for the packets from all the classes active on its level of the hierarchy. There is one self feed per level, each consisting of 8 self slots that represent priorities.
- **self slot** - an element of a self feed that corresponds to each particular priority. There is one self slot per priority per level. All classes, active at the same level, having the same priority are attached to one self slot that they are using to send packets out through.
- **active class (at a particular level)** - a class that is attached to a self slot at the given level.

- **inner feed** - similar to a self feed object, which consists of inner self slots, present on each inner class. There is one inner feed per inner class.
- **inner feed slot** - similar to self slot. Each inner feed consists of inner slots which represent a priority.

Each class has a parent and may have one or more children. Classes that do not have children, are put at level 0, where queues are maintained, and are called 'leaf classes'.
Each class in the hierarchy can prioritize and shape traffic. There are two main parameters, which refer to shaping and one - to prioritizing:

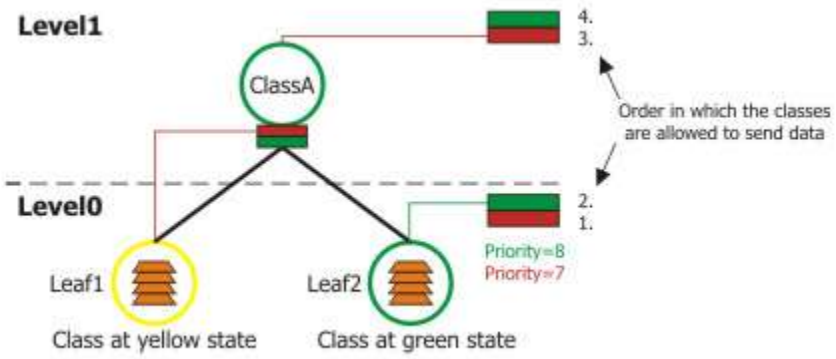
- **limit-at** - normal data rate that is guaranteed to a class (CIR)
- **max-limit** - maximal data rate that is allowed for a class to reach (MIR)
- **priority** - order in which classes are served at the same level (8 is the lowest priority, 1 is the highest)

Each HTB class can be in one of 3 states, depending on data rate that it consumes:

- **green** - a class the actual rate of which is equal or less than **limit-at**. At this state, the class is attached to self slot at the corresponding priority at its level, and is allowed to satisfy its CIR limitation regardless of what limitations its parents have. For example, if we have a leaf class with **limit-at=512000** and its parent has **max-limit=limit-at=128000**, the class will still get its 512kbps! All CIRs of a particular level are satisfied before all MIRs of the same level and any limitations of higher levels.
- **yellow** - a class the actual rate of which is greater than **limit-at** and equal or less than **max-limit** (or **burst-limit** if burst is active). At this state, the class is attached to the inner slot of the corresponding priority of its parent's inner feed, which, in turn, may be attached to either its parent's inner slot of the same priority (in case the parent is also yellow), or to its own level self slot of the same priority (in case the parent is green). Upon the transition to this state, the class 'disconnects' from self feed of its level, and 'connects' to its parent's inner feed.
- **red** - a class the actual rate of which exceeds **max-limit** (or **burst-limit** if burst is active). This class cannot borrow rate from its parent class.

Note: as CIRs are always satisfied before MIRs or other limitations of higher levels are consulted, you should always ensure that the **limit-at** property of any inner class is equal or greater than the sum of all **limit-at** parameter of the children of that inner class.

Priorities
When there is a possibility to send out a packet, HTB queries all its self slots in order of priority, starting with highest priority on the lowest level, till lowest priority on highest level. Each leaf class (packets are stored and enqueued only within qdiscs attached to each leaf class) is ultimately connected to a certain self slot, either directly or through a chain of parent classes:



As you can see from the picture, leaf-classes that are in the green state will always have a higher effective priority than those that are yellow (and, thus, borrowing their rate from parent classes), because their priority is at a lower level (level 0). In this picture, **Leaf1** will be served only after **Leaf2**, although it has a higher priority (priority 7) than **Leaf1** (priority 8).
In case of equal priorities and equal states, HTB serves these classes, using round robin algorithm.

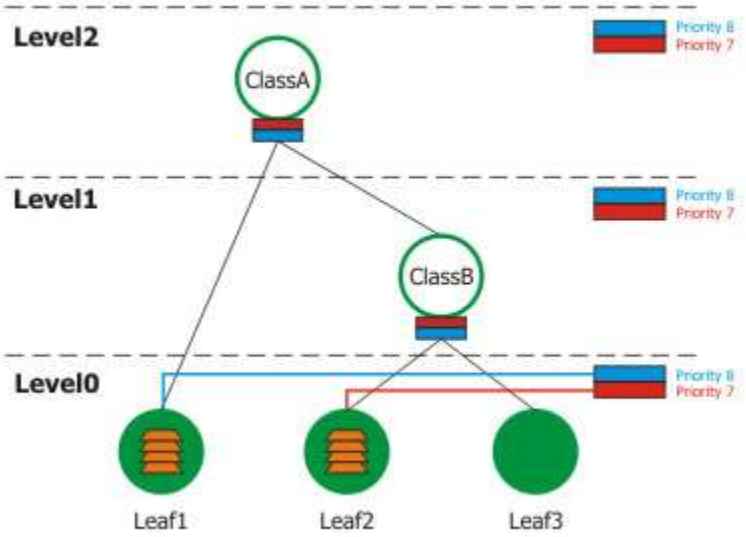
HTB Examples

Here are some examples on how the HTB works.
Imagine the following scenario - we have 3 different kinds of traffic, marked in **/ip firewall mangle** (packet_mark1, packet_mark2 and packet_mark3), and now have bulit a HTB hierarchy:

```
[admin@HauteSpot] queue tree> add name=ClassA parent=Local max-limit=2048000
[admin@HauteSpot] queue tree> add name=ClassB parent=ClassA max-limit=1024000
[admin@HauteSpot] queue tree> add name=Leaf1 parent=ClassA max-limit=2048000 \
\... limit-at=1024000 packet-mark=packet_mark1 priority=8
[admin@HauteSpot] queue tree> add name=Leaf2 parent=ClassB max-limit=1024000 \
\... limit-at=256000 packet-mark=packet_mark2 priority=7
[admin@HauteSpot] queue tree> add name=Leaf3 parent=ClassB max-limit=1024000 \
\... limit-at=768000 packet-mark=packet_mark3 priority=8
[admin@HauteSpot] queue tree> print
Flags: X - disabled, I - invalid
 0   name="ClassA" parent=Local packet-mark="" limit-at=0 queue=default
     priority=8 max-limit=2048000 burst-limit=0 burst-threshold=0
     burst-time=0s

 1   name="ClassB" parent=ClassA packet-mark="" limit-at=0 queue=default
     priority=8 max-limit=1024000 burst-limit=0 burst-threshold=0
     burst-time=0s

 2   name="Leaf1" parent=ClassA packet-mark=packet_mark1 limit-at=1024000
     queue=default priority=8 max-limit=2048000 burst-limit=0
     burst-threshold=0 burst-time=0s

 3   name="Leaf2" parent=ClassB packet-mark=packet_mark2 limit-at=256000
     queue=default priority=7 max-limit=1024000 burst-limit=0
     burst-threshold=0 burst-time=0s

 4   name="Leaf3" parent=ClassB packet-mark=packet_mark3 limit-at=768000
     queue=default priority=8 max-limit=1024000 burst-limit=0
     burst-threshold=0 burst-time=0s
[admin@HauteSpot] queue tree>
```

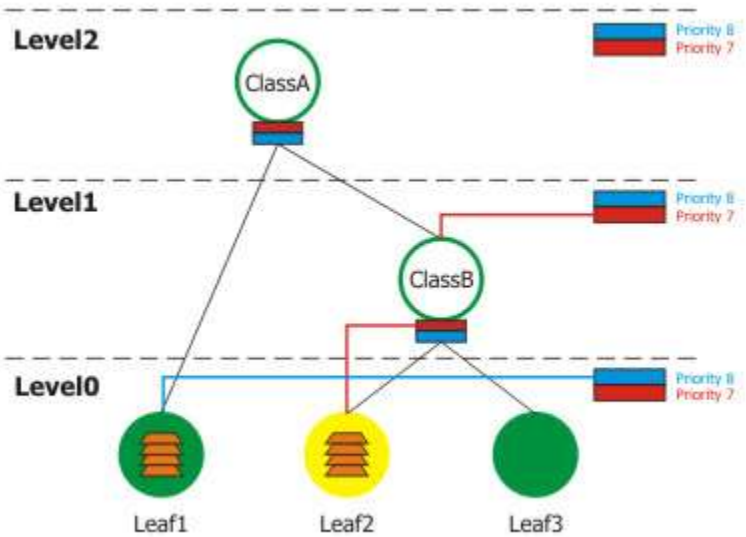
Now let us describe some scenarios, using this HTB hierarchy.

1. Imagine a situation when packets have arrived at Leaf1 and Leaf2. Because of this, Leaf1 attaches itself to this level's (Level 0) self slot with priority=8 and Leaf2 attaches to self slot with priority=7. Leaf3 has nothing to send, so it does nothing.



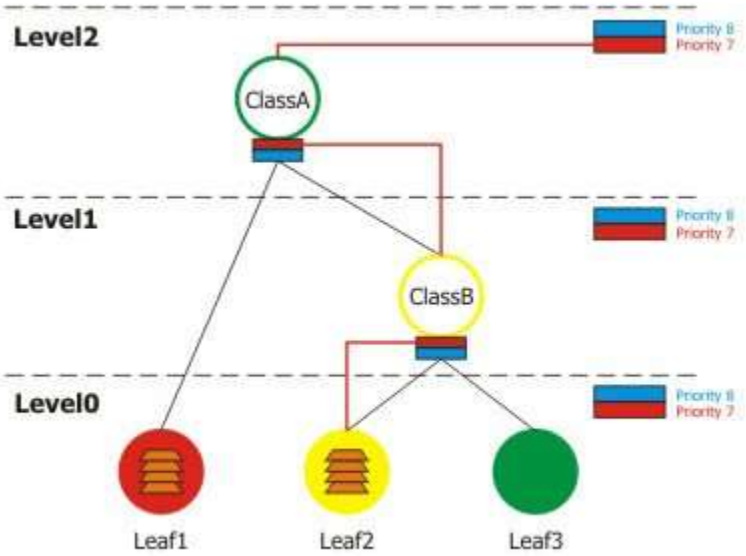
This is a simple situation: there are two active classes (Leaf1 and Leaf2) at Level 0, and as they both are in green state, they are processed in order of their priorities - at first, we serve Leaf2, then Leaf1.

2. Now assume that Leaf2 has to send more than 256kbps, so it needs to go over it's green limit. With the state change, it attaches itself to its parent's (ClassB) inner feed, which recursively attaches itself to Level1 self slot at priority 7. Leaf1 remains in green state - it has packets to send, but their rate is lower than 1Mbps. Leaf3 still has nothing to send.



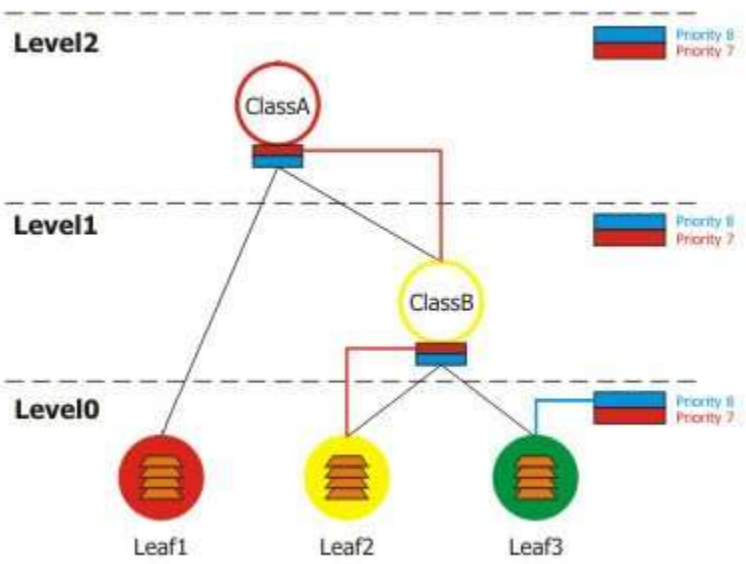
It is very important to understand that Leaf1 now has higher effective priority than Leaf2 (when it is in green state), although we have configured it for a lower priority (8) than Leaf2. It is because Leaf2 has disconnected itself from self feed at Level 0 and is now borrowing rate from its parent (ClassB), which, in turn, has attached to a self feed at Level 1. Thus, the priority of Leaf2 has jumped to Level1. Remember that lowest level is served first, than the next level, and so on, satisfying the attached classes in order of their priority.

3. Consider that Leaf1 has reached its max-limit and changed its state to red, and Leaf2 now uses more than 1Mbps (and less than 2Mbps), so its parent ClassB has to borrow from ClassA and becomes yellow. Leaf3 still has no packets to send.



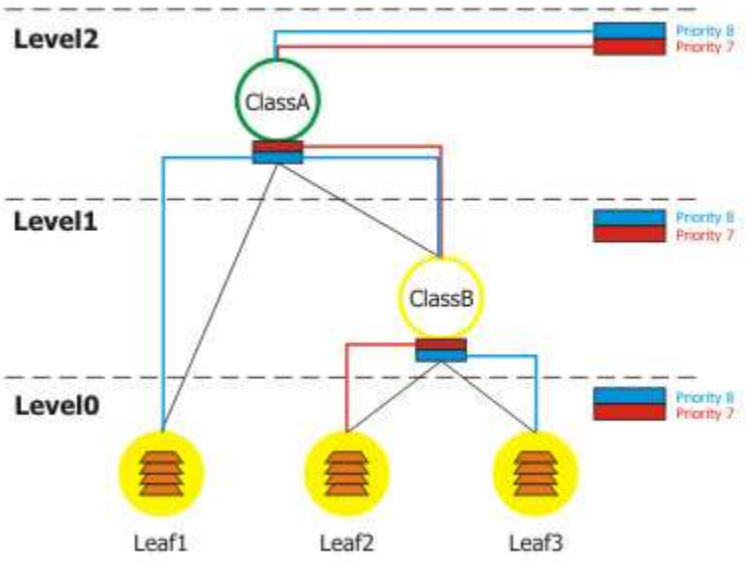
This scenario shows that Leaf1 has reached its max-limit and cannot even borrow from its parent (ClassA), so it is detached from all self slots and inner slots. Leaf2 has recursively reached Level 2, as it borrows from ClassB which, in turn, borrows from ClassA, as it does not have enough rate available. As Leaf3 has no packets to send, the only class that sends is Leaf2.

4. Assume that ClassA reaches its max-limit (2Mbps), so neither ClassB, nor Leaf2 can send as they only rely on borrowing rate, which is impossible as ClassA cannot send. But now, Leaf3 has some packets to send:



In this situation Leaf2 is in yellow state, but it cannot borrow (as Class B cannot borrow from Class A) and Leaf3 is the only class that can send. Note that even though no other classes, including its parents is able to send, Leaf3 can send perfectly well while is is attached to the Level 0 self feed.

5. Finally, let's see what happens, if Leaf1, Leaf2, Leaf3 and ClassB are in the yellow state, and ClassA is green.



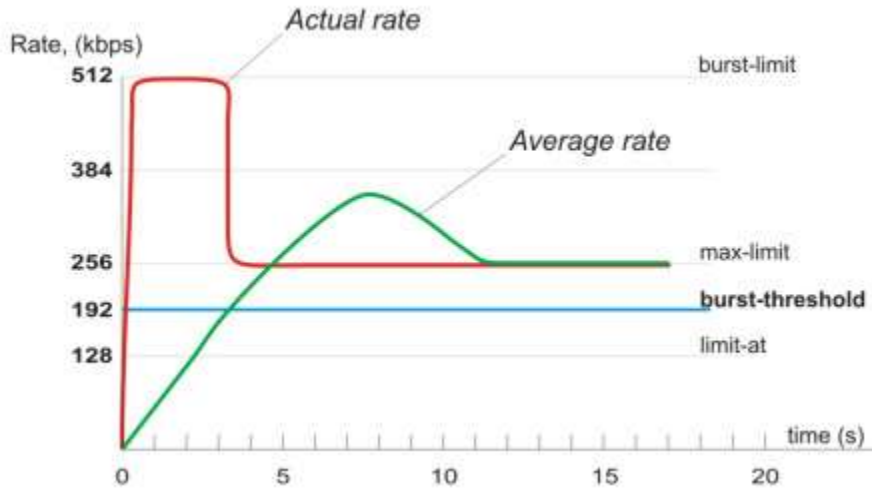
Leaf1 borrows from ClassA, Leaf2 and Leaf3 - from ClassB, and ClassB, in turn, borrows from ClassA. Now all the priorities have 'moved' to Level 2. So Leaf2 is on the highest priority and is served first. As Leaf1 and Leaf3 are of the same priority (8) on the same level (2), they are served using round robin algorithm.

Bursts

Bursts are used to allow higher data rates for a short period of time. Every 1/16 part of the **burst-time**, the router calculates the average data rate of each class over the last **burst-time** seconds. If this average data rate is less than **burst-threshold**, burst is enabled and the effective rate limit (transition to the red state) is set to **burst-limit** bps, otherwise the effective maximal limit falls to

max-limit.

Let us consider the following setup: **max-limit**=256000, **burst-time**=8, **burst-threshold**=192000 and **burst-limit**=512000. When a user is starting to download a file via HTTP, we can observe such situation:



At the beginning the average data rate over the past 8 seconds is 0bps because no traffic has passed through this ruke before it has been created. Since this average data rate is less than **burst-threshold** (192kbps), burst is allowed. After the first second, the average data rate is (0+0+0+0+0+0+0+512)/8=64kbps, which is less than **burst-threshold**. After the second second, average data rate is (0+0+0+0+0+0+512+512)/8=128kbps. After the third second comes the breakpoint when the average data rate becomes larger than **burst-threshold**. At this moment burst is disabled and the effective data rate limitation falls down to **max-limit** (256kbps).

Note how the **burst-time** was used. The actual duration of burst does not depend of **burst-time** alone! It also depends on the **burst-threshold**/**burst-limit** ratio and the actual data rate passing through the bursty class. In this example the burst ratio was 192000/512000=3/8, the time was 8, and the queue has been trying to utilize all available rate the class was providing, so the burst was 3 seconds long.

Now you can easily see why the **burst-threshold** should be between **limit-at** and **max-limit** for normal operation. If you specify **burst-threshold** higher than **max-limit**, then the average rate will tend to **burst-threshold**, but the effective maximal limit will jump between **max-limit** and **burst-limit** constantly (depending on the actual traffic rate, it may happen even on each evaluation point (1/16th of **burst-time**)).

HTB in RouterOS

In addition to interface queues (one queue or HTB tree per interface), 3 virtual 4 HTB trees maintained by RouterOS:

- global-in
- global-total
- global-out

When adding a simple queue, it creates 3 HTB classes (in global-in, global-total and global-out), but it does not add any classes in interface queue. Queue tree is more flexible - you can add it to any of these HTB's.

When packet travels through the router, it passes 4 HTB trees - global-in, global-total, global-out and output interface queue. If it is directed to the router, it passes global-in and global-total HTB queues. If packets are sent from the router, they go through global-total, global-out and output interface queues

Additional Resources

- http://linux-ip.net/articles/Traffic-Control-HOWTO/overview.html
- http://luxik.cdi.cz/~devik/qos/htb/
- http://www.docum.org/docum.org/docs/

Queue Types

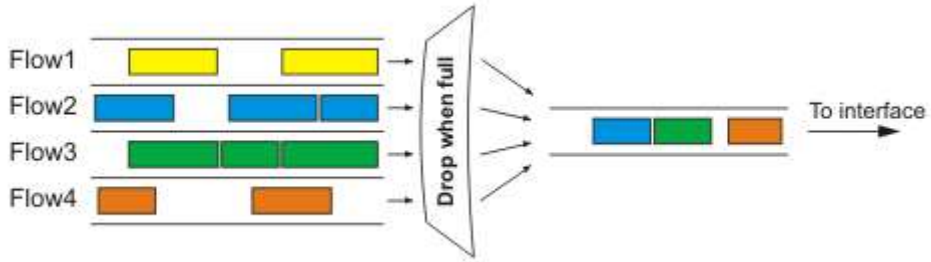
Submenu level: ***/queue type***

Description

You can create your custom queue types in this submenu. Afterwards, you will be able to use them in **/queue tree**, **/queue simple** or **/queue interface**. Note that these queuing disciplines can not limit data rate at all (except for PCQ) - they only reorganize (schedule) packets and drop excess ones (if the queue is getting too long and the managing class can not send the packets quickly enough), so you won't find any rate limitation parameters here (except for PCQ) - only storage limits. Note also that the scheduling is only taking place when the packets are being enqueued in the qdisc, and this only happens when the packets are coming in at the rate faster than the managing class can provide (so this is only a buffer). There are 5 kinds of qdiscs that can be used for storing packets:

PFIFO and BFIFO

These queuing disciplines are based on the FIFO algorithm (First-In First-Out). The difference between PFIFO and BFIFO is that one is measured in packets and the other one in bytes. There is only one parameter called **pfifo-limit** (or **bfifo-limit** in case of BFIFO) which defines how much data a FIFO queue can hold. Every packet that cannot be enqueued (if the queue is full), is dropped. Large queue sizes can increase latency, but utilize channel better.

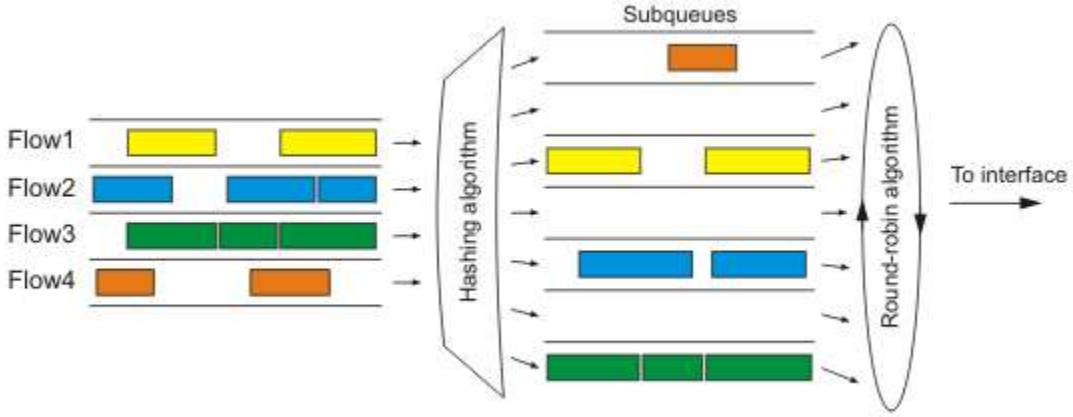


Use FIFO queuing disciplines if you have a non-congested link.

SFQ

Stochastic Fairness Queuing (SFQ) equalizes traffic flows (TCP sessions or UDP streams) when the link is completely full.

The fairness of SFQ is ensured by hashing and round-robin algorithms. Hashing algorithm divides the session traffic over a limited number of subqueues. A traffic flow may be uniquely identified by a tuple (src-address, dst-address, src-port and dst-port), so these parameters are used by SFQ hashing algorithm to classify packets into subqueues.



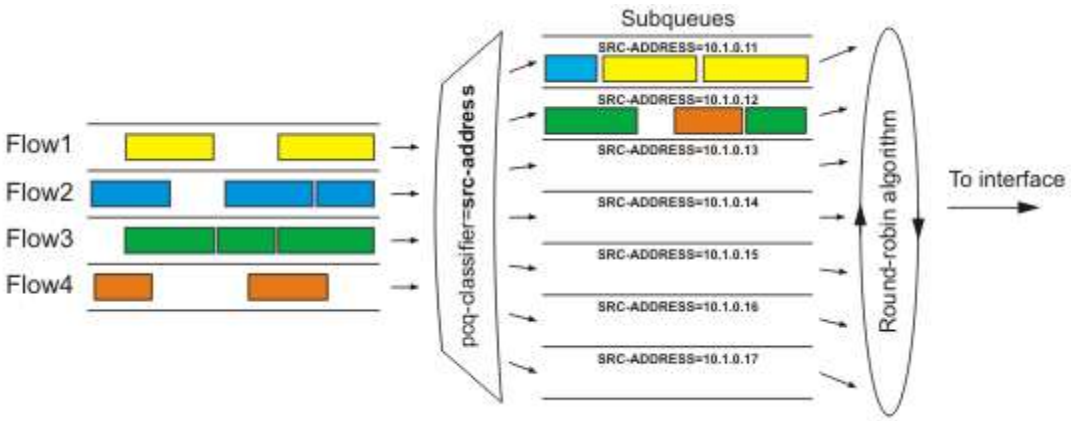
The whole SFQ queue can contain 128 packets and there are 1024 subqueues available for these packets. Each packet stored in a FIFO-like 128 packet buffer, belongs to a certain subqueue, determined by the hash function (a simple function of the tuple values with 10-bit output is used, hence the amount of subqueues is 1024). Stochastic nature of the queueing discipline is observed in that packets from an unpredictable number of flows may actually be classified in the same subqueue. After **sfq-perturb** seconds the hashing algorithm changes and divides the session traffic to other subqueues, so that no separate data flows will be associated with the same subqueue for a long time. The round-robin algorithm dequeues **pcq-allot** bytes from each subqueue in a turn.

Use SFQ for congested links to ensure that connections do not starve. SFQ is especially beneficial on wireless links.

PCQ

To solve some SFQ imperfectness, Per Connection Queuing (PCQ) was created. It is the only classless queuing type in RouterOS that can do rate limitation. It is an improved version of SFQ without its stochastic nature. PCQ also creates subqueues, based on the **pcq-classifier** parameter. Each subqueue has a data rate limit of **pcq-rate** and size of **pcq-limit** packets. The total size of a PCQ queue cannot be greater than **pcq-total-limit** packets.

The following example demonstrates the usage of PCQ with packets, classified by their source address.



If you classify the packets by **src-address** then all packets with different source IP addresses will be grouped into different subqueues. Now you can do the limitation or equalization for each subqueue with the **pcq-rate** parameter. Perhaps, the most significant part is to decide to which interface should we attach this queue. If we will attach it to the Local interface, all traffic from the Public interface will be grouped by src-address (probably it's not what we want), but if we attach it to the Public interface, all traffic from our clients will be grouped by src-address - so we can easily limit or equalize upload for clients. Same can be done for downloads, but in that case **dst-address** classifier will be used, and PCQ put on the local interface.

To equalize rate among subqueues, classified by the **pcq-classifier**, set the **pcq-rate** to **0**! PCQ can be used to dynamically equalize or shape traffic for multiple users, using little administration. In fact, PCQ always equalizes the subqueues, so the **pcq-rate** is just a cap for equalization - a subqueue may get smaller rate, but will never get higher rate.

RED

Random Early Detection (also known as Random Early Drop, as this is how it actually works) is a queuing mechanism which tries to avoid network congestion by controlling the average queue size. When the average queue size reaches **red-min-threshold**, RED starts to drop packets randomly with linearly increasing probability as the average queue size grows up until the average queue size reaches the **red-max-threshold**. The effective queue size at any moment could be higher than the **red-max-threshold** as the probability does not grow very fast, so it is possible to specify a hard limit for the queue size. When the average queue size reaches **red-max-threshold** or becomes larger, all further packets are dropped until the average queue size does not drop below this values (at which point probability calculations will be activated again). The average queue size **avg** is *(1-W)*avg+W*q*, where

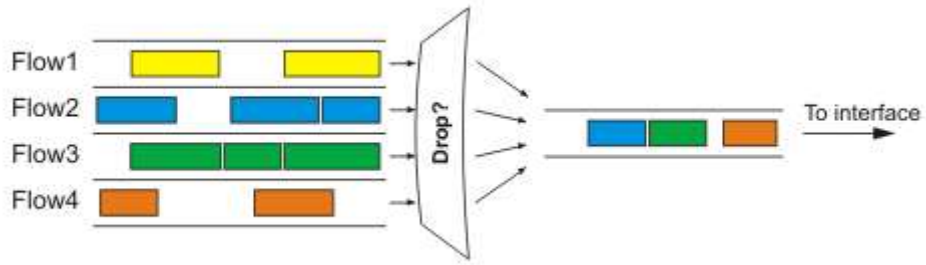
- **q** - current queue length
- **W** - queue weight defined as *burst+1-min=(1-(1-W)^burst)/W*. Note that *log(W)* value ir rounded to integer (so **W** can be 1, 0.1, 0.01, etc.). It is determined experimentally that in many generic cases, **W** is near to min/10*burst

The **pb** probability value is increasing linearly from 0% to 2% as the average queue size grows from **red-min-threshold** to **red-max-threshold**: *pb=0.02*(avg-min)/(max-min)*.

The packet dropping probability **pb** is increasing with **pb** and with enqueued packet count since the last packet was dropped: *pa=pb/(1-count*pb)*.

It is defined experimentally that a good **red-burst** value is *(min+2*max)/3*. And a good **red-max-threshold** is twice **red-min-threshold**.

Note that in the formulas above, **min** means **red-min-threshold**, **max** means **red-max-threshold** and **burst** means **red-burst**.



Used on congested links with high data rates, as it is fast and TCP-friendly.

Property Description

bfifo-limit (*integer*; default: **15000**) - maximum number of bytes that the BFIFO queue can hold
kind (bfifo | pcq | pfifo | red | sfq) - which queuing discipline to use
bfifo - Bytes First-In, First-Out
pcq - Per Connection Queue
pfifo - Packets First-In, First-Out
red - Random Early Detection
sfq - Stochastic Fairness Queuing
name (*name*) - reference name of the queue type
pcq-classifier (dst-address | dst-port | src-address | src-port; default: **""**) - list classifiers for grouping packets into PCQ subqueues. Several classifiers can be used at once, e.g., src-address,src-port will group all packets with different source address and source-ports into separate subqueues
pcq-limit (*integer*; default: **50**) - number of packets that a single PCQ sub-queue can hold
pcq-rate (*integer*; default: **0**) - maximal data rate allowed for each PCQ sub-queue. This is a rate cap, as the subqueues will be equalized anyway
0 - no limitation set (only equalize rates between subqueues)
pcq-total-limit (*integer*; default: **2000**) - number of packets that the whole PCQ queue can hold
pfifo-limit (*integer*) - maximum number of packets that the PFIFO queue can hold
red-avg-packet (*integer*; default: **1000**) - average packet size, used for tuning average queue recalculation time
red-burst (*integer*) - a measure of how fast the average queue size will be influenced by the real queue size, given in bytes. Larger values will smooth the changes, so longer bursts will be allowed
red-limit (*integer*) - hard limit on queue size in bytes. If the real queue size (not average) exceeds this value then all further packets will be discarded until the queue size drops below. This should be higher than **red-max-threshold**+**red-burst**
red-max-threshold (*integer*) - upper limit for average queue size, in bytes. When the size reaches this value, all further packets shall be dropped
red-min-threshold (*integer*) - lower limit for average queue size, in bytes. When the size reaches this value, RED starts to drop packets randomly with a calculated probability
sfq-allot (*integer*; default: **1514**) - amount of bytes that a subqueue is allowed to send before the next subqueue gets a turn (amount of bytes which can be sent from a subqueue in a single round-robin turn), should be at least **1514** for links with 1500 byte MTU
sfq-perturb (*integer*; default: **5**) - how often to shake (perturb) SFQ's hashing algorithm, in seconds

Interface Default Queues

Submenu level: ***/queue interface***

Description

In order to send packets over an interface, they have to be enqueued in a queue even if you do not want to limit traffic at all. Here you can specify the queue type which will be used for transmitting data.

Note that once you configure tree queues for a listed interface, the interface default queue is no longer active for that particular interface, so you need to make sure all packets that goes out through this interface are filtered into some qdiscs inside the HTB tree. Otherwise the packets that are not filtered, are sent out directly (at effective higher priority than any of the packets in the HTB tree), and un-buffered, which ultimately lead to suboptimal performance.

Property Description

interface (*read-only: name*) - name of the interface
queue (*name*; default: **default**) - queue type which will be used for the interface

Example

Set the wireless interface to use **wireless-default** queue:

```
[admin@HauteSpot] queue interface> set 0 queue=wireless-default
[admin@HauteSpot] queue interface> print
 # INTERFACE QUEUE
 0 wlan1     wireless-default
[admin@HauteSpot] queue interface>
```

Simple Queues

Description

The simplest way to limit data rate for specific IP addresses and/or subnets, is to use simple queues.

You can also use simple queues to build advanced QoS applications. They have useful integrated features:

- Peer-to-peer traffic queuing
- Applying queue rules on chosen time intervals
- Priorities
- Using multiple packet marks from */ip firewall mangle*
- Shaping of bidirectional traffic (one limit for the total of upload + download)

Property Description

burst-limit (*integer*/*integer*) - maximum data rate which can be reached while the burst is active, in form of in/out (target upload/download)
burst-threshold (*integer*/*integer*) - average data rate limit, until which the burst is allowed. If the average data rate over the last burst-time seconds is less than **burst-threshold**, the actual data rate may reach **burst-limit**. Otherwise the hard limit is reset to **max-limit**. Set in form of in/out (target upload/download)
burst-time (*integer*/*integer*) - period of time, in seconds, over which the average data rate is calculated, in form of in/out (target upload/download)
direction (none both upload download) - traffic flow directions from the targets' point of view, affected by this queue

none - the queue is effectively inactive
both - the queue limits both target upload and target download
upload - the queue limits only target upload, leaving the download rates unlimited
download - the queue limits only target download, leaving the upload rates unlimited
dst-address (*IP address*/*netmask*) - destination address to match
dst-netmask (*netmask*) - netmask for *dst-address*
interface (*text*) - interface, this queue applies to (i.e., the interface the target is connected to)
limit-at (*integer*/*integer*) - CIR, in form of in/out (target upload/download)
max-limit (*integer*/*integer*) - MIR (in case burst is not active), in form of in/out (target upload/download)
name (*text*) - descriptive name of the queue
p2p (all-p2p | bit-torrent | blubster | direct-connect | edonkey | fasttrack | gnutella | soulseek | winmx) - which type of P2P traffic to match
all-p2p - match all P2P traffic
packet-marks (*multiple choice: name*; default: **""**) - list of packet marks (set by **/ip firewall mangle**) to match. Multiple packet marks are separated by commas (",")
parent (*name*) - name of the parent queue in the hierarchy. Can only be another simple queue
priority (*integer*: 1..8) - priority of the queue. 1 is the highest, 8 - the lowest
queue (*name*/*name*; default: **default/default**) - name of the queue from **/queue type**, in form of in/out
target-addresses (*multiple choice: IP address*/*netmask*) - limitation target IP addresses (source addresses). Multiple addresses are separated by commas
time (*time-time*,sat | fri | thu | wed | tue | mon | sun{+}; default: **""**) - limit queue effect to a specified time period
total-burst-limit (*integer*) - burst limit for *global-total* (cumulative rate, upload + download) queue
total-burst-threshold (*integer*) - burst threshold for *global-total* (cumulative rate, upload + download) queue
total-burst-time (*time*) - burst time for *global-total* queue
total-limit-at (*integer*) - limit-at for *global-total* (cumulative rate, upload + download) queue
total-max-limit (*integer*) - max-limit for *global-total* (cumulative rate, upload + download) queue
total-queue (*name*) - queuing discipline to use for **global-total** queue

Queue Trees

Submenu level: ***/queue tree***

Description

The queue trees should be used when you want to use sophisticated data rate allocation based on protocols, ports, groups of IP addresses, etc. At first you have to mark packet flows with a mark under **/ip firewall mangle** and then use this mark as an identifier for packet flows in queue trees.

Property Description

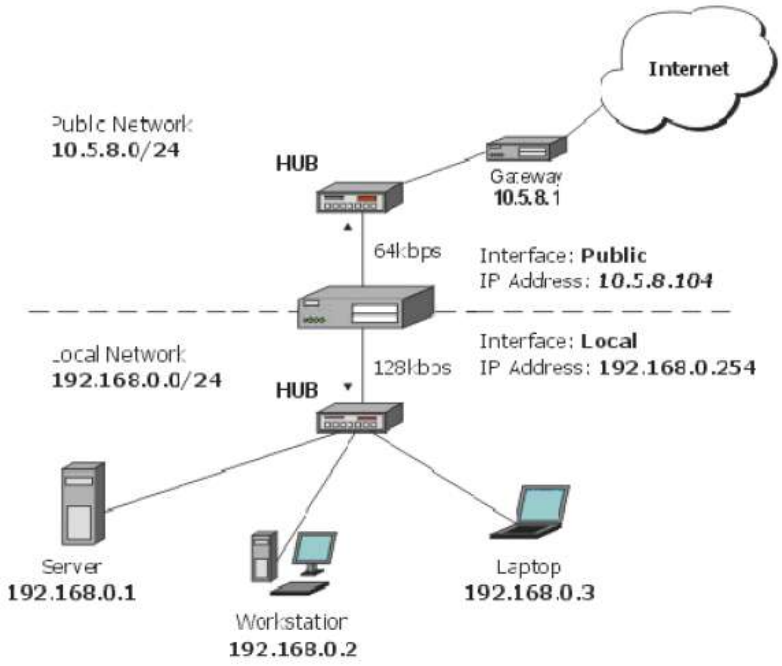
burst-limit (*integer*) - maximum data rate which can be reached while the burst is active
burst-threshold (*integer*) - average data rate limit, until which the burst is allowed. If the average data rate over the last burst-time seconds is less than **burst-threshold**, the actual data rate may reach **burst-limit**. Otherwise the hard limit is reset to **max-limit**
burst-time (*time*) - period of time, in seconds, over which the average data rate is calculated
limit-at (*integer*) - CIR
max-limit (*integer*) - MIR (in case burst is not active)
name (*text*) - descriptive name for the queue
packet-mark (*text*) - packet flow mark (set by **/ip firewall mangle**) to match. This creates a filter that puts the packets with the given mark into this queue

parent (*text*) - name of the parent queue. The top-level parents are the available interfaces (actually, main HTB). Lower level parents can be other tree queues
priority (*integer*: 1..8) - priority of the queue. 1 is the highest, 8 - the lowest
queue (*text*) - name of the queue type. Types are defined under **/queue type**

Application Examples

Example of emulating a 128Kibps/64Kibps Line

Assume, we want to emulate a 128Kibps download and 64Kibps upload line, connecting IP network **192.168.0.0/24**. The network is served through the Local interface of customer's router. The basic network setup is in the following diagram:



To solve this situation, we will use simple queues.

IP addresses on HauteSpot router:

```
[admin@HauteSpot] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
 #   ADDRESS            NETWORK         BROADCAST       INTERFACE
 0   192.168.0.254/24   192.168.0.0     192.168.0.255   Local
 1   10.5.8.104/24      10.5.8.0        10.5.8.255      Public
[admin@HauteSpot] ip address>
```

And routes:

```
[admin@HauteSpot] ip route> print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        G GATEWAY         DIS INTE...
 0 A S  0.0.0.0/0                          r 10.5.8.1        1   Public
 1 ADC  10.5.8.0/24        10.5.8.104                        0   Public
 2 ADC  192.168.0.0/24     192.168.0.254                     0   Local
[admin@HauteSpot] ip route>
```

Add a simple queue rule, which will limit the download traffic to 128Kib/s and upload to 64Kib/s for clients on the network **192.168.0.0/24**, served by the interface **Local**:

```
[admin@HauteSpot] queue simple> add name=Limit-Local interface=Local \
\... target-address=192.168.0.0/24 max-limit=65536/131072
[admin@HauteSpot] queue simple> print
Flags: X - disabled, I - invalid, D - dynamic
 0    name="Limit-Local" target-addresses=192.168.0.0/24 dst-address=0.0.0.0/0
      interface=Local parent=none priority=8 queue=default/default
      limit-at=0/0 max-limit=65536/131072 total-queue=default
[admin@HauteSpot] queue simple>
```

The **max-limit** parameter cuts down the maximum available bandwidth. From the clients' point of view, the value **65536/131072** means that they will get maximum of 131072bps for download and 65536bps for upload. The **target-addresses** parameter defines the target network (or networks, separated by a comma) to which the queue rule will be applied.

Now see the traffic load:

```
[admin@HauteSpot] interface> monitor-traffic Local
  received-packets-per-second: 7
       received-bits-per-second: 68kbps
        sent-packets-per-second: 13
           sent-bits-per-second: 135kbps

[admin@HauteSpot] interface>
```

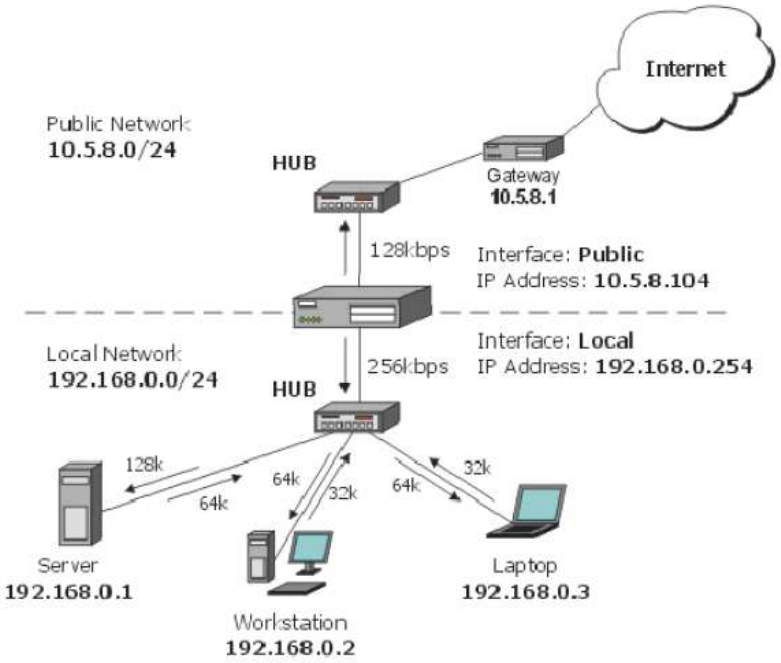
Probably, you want to exclude the server from being limited, if so, add a queue for it without any limitation (**max-limit=0/0** which means no limitation) and move it to the beginning of the list:

```
[admin@HauteSpot] queue simple> add name=Server target-addresses=192.168.0.1/32 \
\... interface=Local
[admin@HauteSpot] queue simple> print
Flags: X - disabled, I - invalid, D - dynamic
 0    name="Limit-Local" target-addresses=192.168.0.0/24 dst-address=0.0.0.0/0
      interface=Local parent=none priority=8 queue=default/default
      limit-at=0/0 max-limit=65536/131072 total-queue=default

 1    name="Server" target-addresses=192.168.0.1/32 dst-address=0.0.0.0/0
      interface=Local parent=none priority=8 queue=default/default
      limit-at=0/0 max-limit=0/0 total-queue=default
[admin@HauteSpot] queue simple> mo 1 0
[admin@HauteSpot] queue simple> print
Flags: X - disabled, I - invalid, D - dynamic
 0    name="Server" target-addresses=192.168.0.1/32 dst-address=0.0.0.0/0
      interface=Local parent=none priority=8 queue=default/default
      limit-at=0/0 max-limit=0/0 total-queue=default

 1    name="Limit-Local" target-addresses=192.168.0.0/24 dst-address=0.0.0.0/0
      interface=Local parent=none priority=8 queue=default/default
      limit-at=0/0 max-limit=65536/131072 total-queue=default
[admin@HauteSpot] queue simple>
```

Queue Tree Example With Masquerading

In the previous example we dedicated 128Kib/s download and 64Kib/s upload traffic for the local network. In this example we will guarantee 256Kib/s download (128Kib/s for the server, 64Kib/s for the Workstation and also 64Kib/s for the Laptop) and 128Kib/s for upload (64/32/32Kib/s, respectivelly) for local network devices. Additionally, if there is spare bandwidth, share it among users equally. For example, if we turn off the laptop, share its 64Kib/s download and 32Kib/s upload to the Server and Workstation.

When using masquerading, you have to mark the outgoing connection with **new-connection-mark** and take the **mark-connection** action. When it is done, you can mark all packets which belong to this connection with the **new-packet-mark** and use the **mark-packet** action.



1. At first, mark the Server's download and upload traffic. With the first rule we will mark the outgoing connection and with the second one, all packets, which belong to this connection:

```
[admin@HauteSpot] ip firewall mangle> add src-address=192.168.0.1/32 \
\... action=mark-connection new-connection-mark=server-con chain=prerouting
[admin@HauteSpot] ip firewall mangle> add connection-mark=server-con \
\... action=mark-packet new-packet-mark=server chain=prerouting
[admin@HauteSpot] ip firewall mangle> print
Flags: X - disabled, I - invalid, D - dynamic
 0   chain=prerouting src-address=192.168.0.1 action=mark-connection
     new-connection-mark=server-con

 1   chain=prerouting connection-mark=server-con action=mark-packet
     new-packet-mark=server
[admin@HauteSpot] ip firewall mangle>
```

2. The same for Laptop and Workstation:

```
[admin@HauteSpot] ip firewall mangle> add src-address=192.168.0.2 \
\... action=mark-connection new-connection-mark=lap_works-con chain=prerouting
[admin@HauteSpot] ip firewall mangle> add src-address=192.168.0.3 \
\... action=mark-connection new-connection-mark=lap_works-con chain=prerouting
[admin@HauteSpot] ip firewall mangle> add connection-mark=lap_works-con \
\... action=mark-packet new-packet-mark=lap_work chain=prerouting
[admin@HauteSpot] ip firewall mangle> print
Flags: X - disabled, I - invalid, D - dynamic
 0   chain=prerouting src-address=192.168.0.1 action=mark-connection
     new-connection-mark=server-con
```

```
 1   chain=prerouting connection-mark=server-con action=mark-packet
     new-packet-mark=server

 2   chain=prerouting src-address=192.168.0.2 action=mark-connection
     new-connection-mark=lap_works-con

 3   chain=prerouting src-address=192.168.0.3 action=mark-connection
     new-connection-mark=lap_works-con

 4   chain=prerouting connection-mark=lap_works-con action=mark-packet
     new-packet-mark=lap_work
[admin@HauteSpot] ip firewall mangle>
```

As you can see, we marked connections that belong for Laptop and Workstation with the same flow.

3. In **/queue tree** add rules that will limit Server's download and upload:

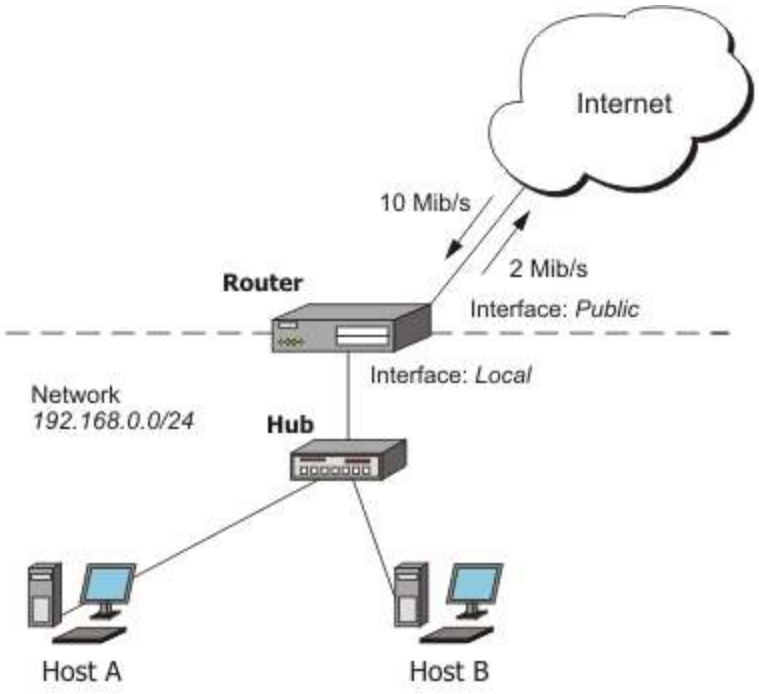
```
[admin@HauteSpot] queue tree> add name=Server-Download parent=Local \
\... limit-at=131072 packet-mark=server max-limit=262144
[admin@HauteSpot] queue tree> add name=Server-Upload parent=Public \
\... limit-at=65536 packet-mark=server max-limit=131072
[admin@HauteSpot] queue tree> print
Flags: X - disabled, I - invalid
 0   name="Server-Download" parent=Local packet-mark=server limit-at=131072
     queue=default priority=8 max-limit=262144 burst-limit=0
     burst-threshold=0 burst-time=0s

 1   name="Server-Upload" parent=Public packet-mark=server limit-at=65536
     queue=default priority=8 max-limit=131072 burst-limit=0
     burst-threshold=0 burst-time=0s
[admin@HauteSpot] queue tree>
```

And similar config for Laptop and Workstation:

```
[admin@HauteSpot] queue tree> add name=Laptop-Wkst-Down parent=Local \
\... packet-mark=lap_work limit-at=65535 max-limit=262144
[admin@HauteSpot] queue tree> add name=Laptop-Wkst-Up parent=Public \
\... packet-mark=lap_work limit-at=32768 max-limit=131072
[admin@HauteSpot] queue tree> print
Flags: X - disabled, I - invalid
 0   name="Server-Download" parent=Local packet-mark=server limit-at=131072
     queue=default priority=8 max-limit=262144 burst-limit=0
     burst-threshold=0 burst-time=0s

 1   name="Server-Upload" parent=Public packet-mark=server limit-at=65536
     queue=default priority=8 max-limit=131072 burst-limit=0
     burst-threshold=0 burst-time=0s

 2   name="Laptop-Wkst-Down" parent=Local packet-mark=lap_work limit-at=65535
     queue=default priority=8 max-limit=262144 burst-limit=0
     burst-threshold=0 burst-time=0s

 3   name="Laptop-Wkst-Up" parent=Public packet-mark=lap_work limit-at=32768
     queue=default priority=8 max-limit=131072 burst-limit=0
     burst-threshold=0 burst-time=0s
[admin@HauteSpot] queue tree>
```

Equal bandwidth sharing among users

This example shows how to equally share 10Mibps download and 2Mbps upload among active users in the network **192.168.0.0/24**. If **Host A** is downloading 2 Mbps, **Host B** gets 8 Mbps and vice versa. There might be situations when both hosts want to use maximum bandwidth (10 Mbps), then they will receive 5 Mbps each, the same goes for upload. This setup is also valid for more than 2 users.



At first, mark all traffic, coming from local network **192.168.0.0/24** with a mark **users**:

```
/ip firewall mangle add chain=forward src-address=192.168.0.0/24 \
   action=mark-connection new-connection-mark=users-con
/ip firewall mangle add connection-mark=users-con action=mark-packet \
   new-packet-mark=users chain=forward
```

Now we will add 2 new PCQ types. The first, called **pcq-download** will group all traffic by destination address. As we will attach this queue type to the **Local** interface, it will create a dynamic queue for each destination address (user) which is downloading to the network **192.168.0.0/24**. The second type, called **pcq-upload** will group the traffic by source address. We will attach this queue to the **Public** interface so it will make one dynamic queue for each user who is uploading to Internet from the local network **192.168.0.0/24**.

```
/queue type add name=pcq-download kind=pcq pcq-classifier=dst-address
/queue type add name=pcq-upload kind=pcq pcq-classifier=src-address
```

Finally, make a queue tree for download traffic:

```
/queue tree add name=Download parent=Local max-limit=10240000
/queue tree add parent=Download queue=pcq-download packet-mark=users
```

And for upload traffic:

```
/queue tree add name=Upload parent=Public max-limit=2048000
/queue tree add parent=Upload queue=pcq-upload packet-mark=users
```

Note! If your ISP cannot guarantee you a fixed amount of traffic, you can use just one queue for upload and one for download, attached directly to the interface:

```
/queue tree add parent=Local queue=pcq-download packet-mark=users
/queue tree add parent=Public queue=pcq-upload packet-mark=users
```

Basic setup of DHCP Client and Server

Document revision: *2.8 (Wednesday, July 22, 2009)*
Applies to: HauteRouterOS V3.0

General Information

Summary

The DHCP (Dynamic Host Configuration Protocol) is needed for easy distribution of IP addresses in a network. The Hautespot RouterOS implementation includes both - server and client parts and is compliant with RFC2131.

General usage of DHCP:

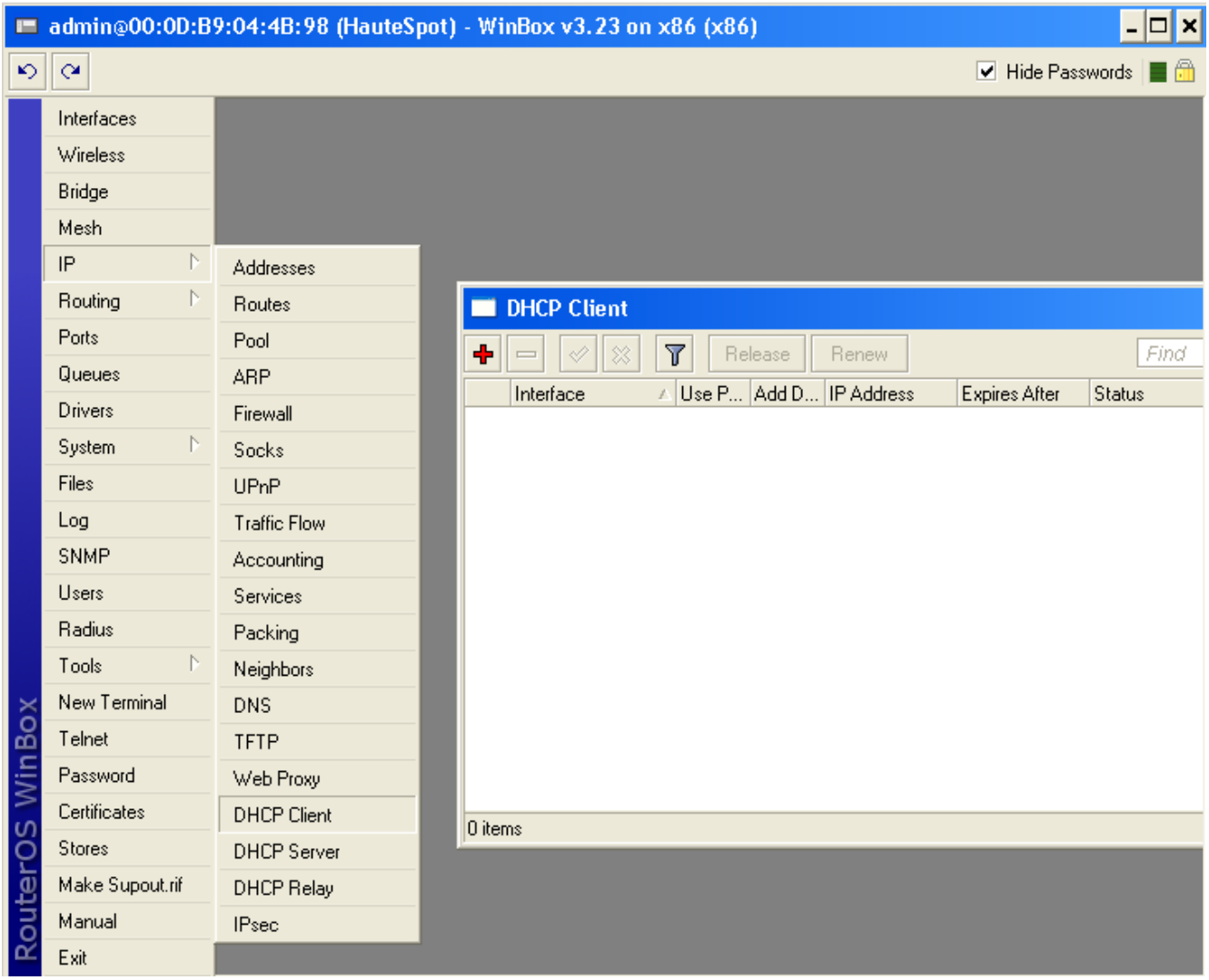
- IP assignment in LAN, cable-modem, and wireless systems
- Obtaining IP settings on cable-modem systems

Quick Setup Guide

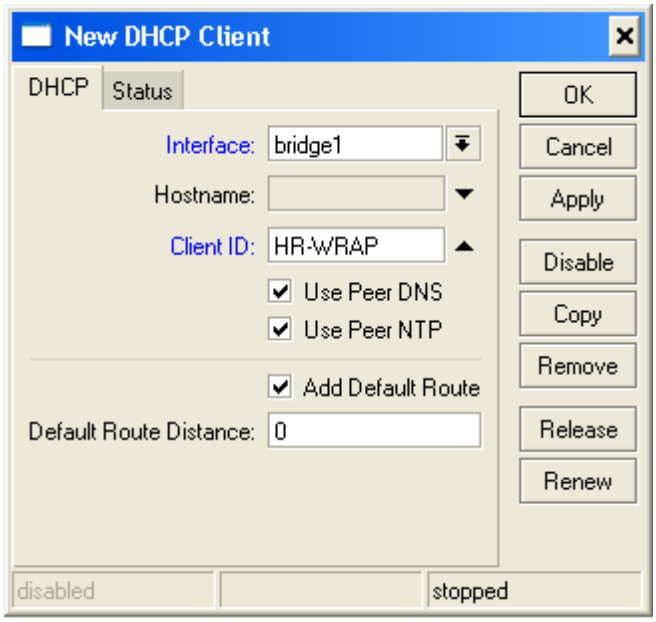
This document will show you how to setup a DHCP-Server and DHCP-Client using the HSConfigurator tool.

DHCP-Client Setup

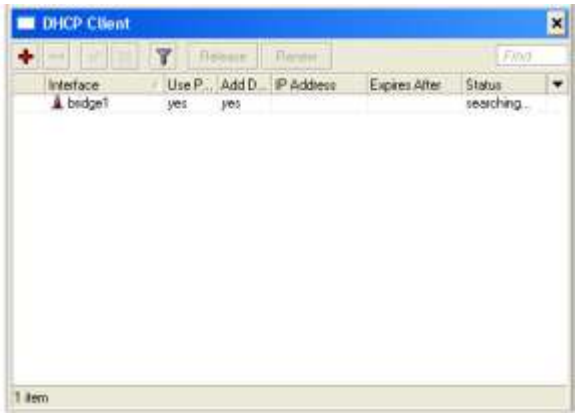
- DHCP Client setup is very simple. In the HSConfigurator tool select the 'IP' menu, this will open a sub menu in which you must select 'DCHP Client'. This will open the DHCP Client menu.



- In the DHCP Client menu you will need to select the button to open the 'New DCHP client' menu. The only change needed in this menu will be to set the interface to bridge1. You may also specify a Client ID and DHCP hostname however neither are required. A Client ID may be useful for network reference. Select 'OK' to add the new DHCP Client.



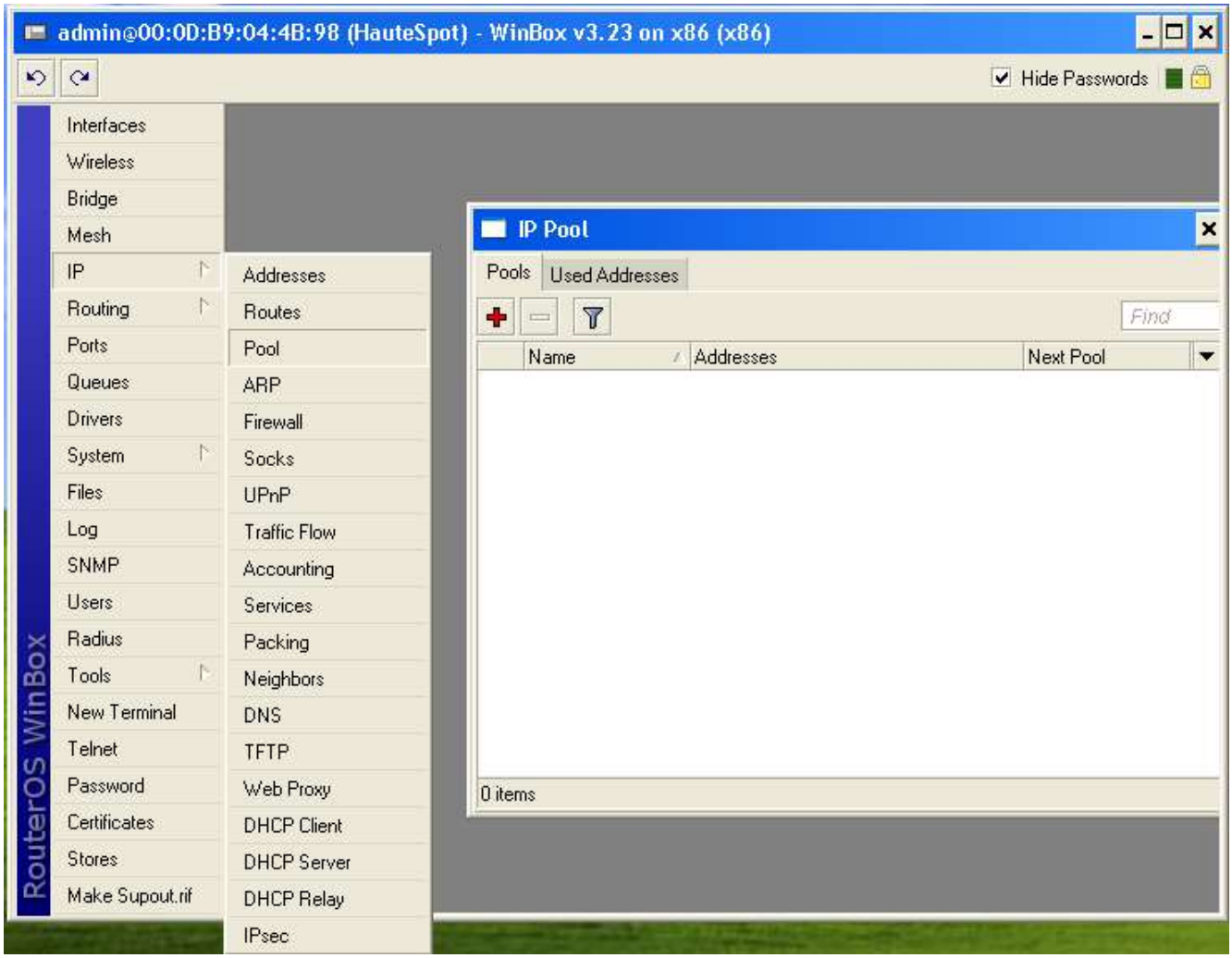
- Once the new DHCP Client has been added you can determine very easily if an address has been acquired. The figure below is an example of a bound interface as opposed to



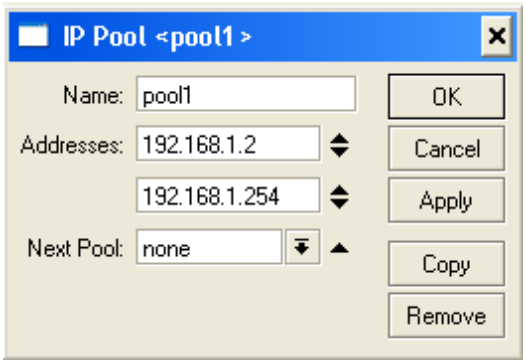
an unbound interface.

DHCP-Server Setup

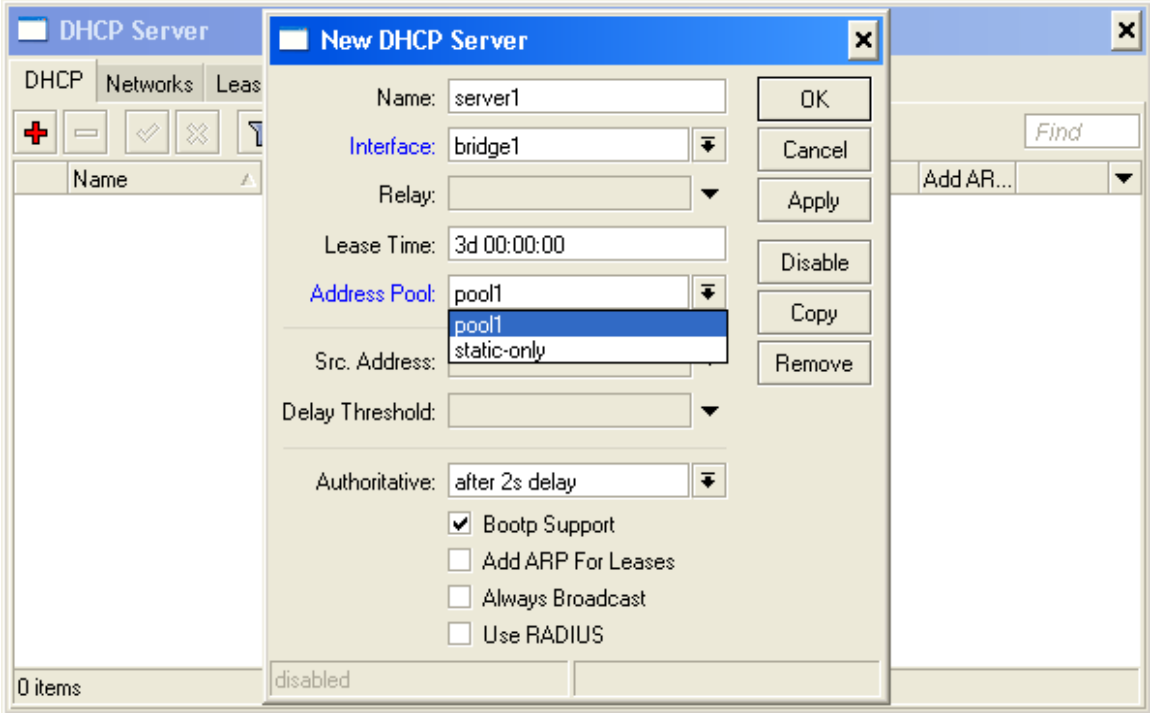
- DHCP-Sever setup requires a little more information than DHCP-Client. In this setup guide most of the information will be assumed. If any of the information appears to be incorrect you may need to make the necessary corrections. To begin open the 'IP' menu and select 'Pool' in the sub menu. This will open the 'IP Pool' menu.



- Select the button to open the 'New IP Pool' window. In this window you will need to enter the starting and ending IP addresses you want to use. By default the HR-WRAP comes with an IP address of 192.168.1.1 so the range of 192.168.1.2-192.168.1.254 was used as an example. We will not be using the 'next pool' feature. Select the 'OK' button in the 'New IP Pool' window once your IP range has been specified.



- Once the pool has been added you can now close the 'IP Pool' menu. We will now need to open the DCHP server menu by select 'IP' and then selecting 'DCHP Server' near the bottom of the sub menu.
- Add a new DHCP-Server entry by selecting the button. In the 'New DHCP Server' menu we will need to change the interface to bridge1 and the address pool to pool1. Select 'OK' to add the entry into the DHCP-Server.



- Once the DHCP-Server has been entered you can use the 'Leases' tab in the 'DHCP Server' menu to see what IP addresses are being leased out.

HauteRouterOS Basic Setup Guide

Document revision: 1.1 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

HauteRouterOS™ is independent Linux-based Operating System for wireless routers. It does not require any additional components and has no software prerequirements. It is designed with easy-to-use yet powerful interface allowing network administrators to deploy network structures and functions, that would require long education elsewhere simply by following the Reference Manual (and even without it).

Description

HauteRouterOS™ is a powerful network router software stack for use on the HR-WRAP platform. Remote control with easy real-time Windows application (HSConfig)

- Advanced Quality of Service control with burst support
- Stateful firewall with P2P protocol filtering, tunnels and IPsec
- STP bridging with filtering capabilities
- WDS and Virtual AP features
- HotSpot for Plug-and-Play access
- RIP, OSPF, BGP routing protocols
- Gigabit Ethernet ready
- V.35, X.21, T1/E1 synchronous support
- async PPP with RADUIS AAA
- IP Telephony
- remote HSConfig GUI admin
- telnet/ssh/serial console admin
- real-time configuration and monitoring
- and much more (please see the Specifications Sheet)

The Guide describes the basic steps of configuring a HR-WRAPDX running HauteRouterOS™.

Logging into the HauteSpot Router

Description

Normally you connect to the router by IP addresses with any telnet or SSH client software (a simple text-mode telnet client is usually called telnet and is distributed together with almost any OS). You can also use graphical configuration tool for Windows (also can be run in Linux using Wine) called HSConfig. To get HSConfig, connect to the router's IP address with a web browser, and follow the link to download HSConfig.exe from the router.

MAC-telnet is used to connect to a router when there is no other way to connect to it remotely if the router has no IP address or in case of misconfigured firewall. MAC-telnet can only be used from the same broadcast domain (so there should be no routers in between) as any of the router's enabled interfaces (you can not connect to a disabled interface). MAC-telnet program is a part of the Neighbor Viewer. Download it from www.hautespot.net, unpack both files contained in the archive to the same directory, and run NeighborViewer.exe. A list of HauteSpot routers working in the same broadcast domain will be showed double-click the one you need to connect to. Note that HSConfig is also able to connect to routers by their MAC addresses, and has the discovery tool built-in.

You can also connect to the router using a standard DB9 serial null-modem cable from any PC. Default settings of the router's serial port are 9600 bits/s, 8 data bits, 1 stop bit, no parity, hardware (RTS/CTS) flow control. Use terminal emulation

program (like HyperTerminal or SecureCRT in Windows, or minicom in UNIX/Linux) to connect to the router. The router will beep twice when booted up, and you should see the login prompt shortly before that (check cabling and serial port settings if you do not see anything in the terminal window).

When logging into the router via terminal console, you will be presented with the HauteRouterOS™ login prompt. Use 'admin' and no password (hit [Enter]) for logging in the router for the first time, for example:

```
HauteSpot v2.9
Login: admin
Password:
```

The password can be changed with the **/password** command.

```
[admin@HauteSpot] > password
old password:
new password: ************
retype new password: ************
[admin@HauteSpot] >
```

Adding Software Packages

Description

The basic installation comes only with the **system** package. This includes basic IP routing and router administration. To have additional features such as IP Telephony, OSPF, wireless and so on, you will need to download additional software packages.

The additional software packages should have **the same version** as the system package. If not, the package won't be installed. Please consult the HauteRouterOS™ Software Package Installation and Upgrading Manual for more detailed information about installing additional software packages.

To upgrade the router packages, simply upload the packages to the router via ftp, using the binary transfer mode. After you have uploaded the packages, reboot the router, and the features that are provided by those packages will be available (regarding your license type, of course).

Navigating The Terminal Console

Description

Welcome Screen and Command Prompt

After logging into the router you will be presented with the HauteRouterOS™ Welcome Screen and command prompt, for example:

```
HauteRouterOS 2.9 (c) 1999-2004       http://www.hautespot.net/




Terminal xterm detected, using multiline input mode
[admin@HauteSpot] >
```

The command prompt shows the identity name of the router and the current menu level, for example:

```
[admin@HauteSpot] >
[admin@HauteSpot] interface>
[admin@HauteSpot] ip address>
```

Commands

The list of available commands at any menu level can be obtained by entering the question mark '?', for example:

```
[admin@HauteSpot] >

log/ -- System logs
quit -- Quit console
radius/ -- Radius client settings
certificate/ -- Certificate management
special-login/ -- Special login users
redo -- Redo previously undone action
driver/ -- Driver management
ping -- Send ICMP Echo packets
setup -- Do basic setup of system
interface/ -- Interface configuration
password -- Change password
undo -- Undo previous action
port/ -- Serial ports
import -- Run exported configuration script
snmp/ -- SNMP settings
user/ -- User management
file/ -- Local router file storage.
system/ -- System information and utilities
queue/ -- Bandwidth management
ip/ -- IP options
tool/ -- Diagnostics tools
ppp/ -- Point to Point Protocol
routing/ -- Various routing protocol settings
export  --

[admin@HauteSpot] >
```

```
[admin@HauteSpot] ip>


.. -- go up to root
service/ -- IP services
socks/ -- SOCKS version 4 proxy
arp/ -- ARP entries management
upnp/ -- Universal Plug and Play
dns/ -- DNS settings
address/ -- Address management
accounting/ -- Traffic accounting
the-proxy/ --
vrrp/ -- Virtual Router Redundancy Protocol
pool/ -- IP address pools
packing/ -- Packet packing settings
neighbor/ -- Neighbors
route/ -- Route management
firewall/ -- Firewall management
dhcp-client/ -- DHCP client settings
dhcp-relay/ -- DHCP relay settings
dhcp-server/ -- DHCP server settings
hotspot/ -- HotSpot management
ipsec/ -- IP security
web-proxy/ -- HTTP proxy
export --

[admin@HauteSpot] ip>
```

The list of available commands and menus has short descriptions next to the items. You can move to the desired menu level by typing its name and hitting the [Enter] key, for example:

```
[admin@HauteSpot] >                  | Base level menu
[admin@HauteSpot] > driver           | Enter 'driver' to move to the driver
                                     | level menu
[admin@HauteSpot] driver> /          | Enter '/' to move to the base level menu
                                     | from any level
[admin@HauteSpot] > interface        | Enter 'interface' to move to the
                                     | interface level menu
[admin@HauteSpot] interface> /ip     | Enter '/ip' to move to the IP level menu
                                     | from any level
[admin@HauteSpot] ip>                |
```

A command or an argument does not need to be completed, if it is not ambiguous. For example, instead of typing **interface** you can type just **in** or **int**. To complete a command use the [Tab] key. Note that the completion is optional, and you can just use short command and parameter names

The commands may be invoked from the menu level, where they are located, by typing its name. If the command is in a different menu level than the current one, then the command should be invoked using its full (absolute) or relative path, for example:

```
[admin@HauteSpot] ip route> print                | Prints the routing table
[admin@HauteSpot] ip route> .. address print     | Prints the IP address table
[admin@HauteSpot] ip route> /ip address print    | Prints the IP address table
```

The commands may have arguments. The arguments have their names and values. Some commands, may have a required argument that has no name.

Summary on executing the commands and navigating the menus

Command	Action
command [Enter]	Executes the command
[?]	Shows the list of all available commands
command [?]	Displays help on the command and the list of arguments
command argument [?]	Displays help on the command's argument
[Tab]	Completes the command/word. If the input is ambiguous, a second [Tab] gives possible options
/	Moves up to the base level
/command	Executes the base level command
..	Moves up one level
""	Specifies an empty string
"word1 word2"	Specifies a string of 2 words that contain a space

You can abbreviate names of levels, commands and arguments.

For the IP address configuration, instead of using the **address** and **netmask** arguments, in most cases you can specify the address together with the number of true bits in the network mask, i.e., there is no need to specify the **netmask** separately. Thus, the following two entries would be equivalent:

```
/ip address add address 10.0.0.1/24 interface ether1
/ip address add address 10.0.0.1 netmask 255.255.255.0 interface ether1
```

Notes

You must specify the size of the network mask in the address argument, even if it is the 32-bit subnet, i.e., use **10.0.0.1/32** for `address=10.0.0.1 netmask=255.255.255.255`

Basic Configuration Tasks

Description

Interface Management

Before configuring the IP addresses and routes please check the **/interface** menu to see the list of available interfaces. If you have Plug-and-Play cards installed in the router, it is most likely that the device drivers have been loaded for them automatically, and the relevant interfaces appear on the **/interface print** list, for example:

```
[admin@HauteSpot] interface> print
Flags: X - disabled, D - dynamic, R - running
 #    NAME                        TYPE            RX-RATE    TX-RATE    MTU
 0  R ether1                      ether           0          0          1500
 1  R ether2                      ether           0          0          1500
 2 X  wavelan1                    wavelan         0          0          1500
 3 X  prism1                      wlan            0          0          1500
[admin@HauteSpot] interface>
```

The interfaces need to be enabled, if you want to use them for communications. Use the **/interface enable name** command to enable the interface with a given name or number, for example:

```
[admin@HauteSpot] interface> print
Flags: X - disabled, D - dynamic, R - running
 #    NAME                        TYPE            RX-RATE    TX-RATE    MTU
 0 X  ether1                      ether           0          0          1500
 1 X  ether2                      ether           0          0          1500
[admin@HauteSpot] interface> enable 0
[admin@HauteSpot] interface> enable ether2
[admin@HauteSpot] interface> print
Flags: X - disabled, D - dynamic, R - running
 #    NAME                        TYPE            RX-RATE    TX-RATE    MTU
 0  R ether1                      ether           0          0          1500
 1  R ether2                      ether           0          0          1500
[admin@HauteSpot] interface>
```

The interface name can be changed to a more descriptive one by using **/interface set** command:

```
[admin@HauteSpot] interface> set 0 name=Local; set 1 name=Public
[admin@HauteSpot] interface> print
Flags: X - disabled, D - dynamic, R - running
 #    NAME                        TYPE            RX-RATE    TX-RATE    MTU
 0  R Local                       ether           0          0          1500
 1  R Public                      ether           0          0          1500
```

```
[admin@HauteSpot] interface>
```

Setup Command

Command name: ***/setup***

Description

The initial setup of the router can be done by using the **/setup** command which offers the following configuration:

- reset all router configuration
- load interface driver
- configure ip address and gateway
- setup dhcp client
- setup dhcp server
- setup pppoe client
- setup pptp client

Configure IP address on router, using the Setup command

Execute the **/setup** command from command line:

```
[admin@HauteSpot] > setup
  Setup uses Safe Mode. It means that all changes that are made during setup
are reverted in case of error, or if [Ctrl]+[C] is used to abort setup. To
keep
changes exit setup using the [X] key.

[Safe Mode taken]
  Choose options by pressing one of the letters in the left column, before
dash. Pressing [X] will exit current menu, pressing Enter key will select the
entry that is marked by an '*'. You can abort setup at any time by pressing
[Ctrl]+[C].
Entries marked by '+' are already configured.
Entries marked by '-' cannot be used yet.
Entries marked by 'X' cannot be used without installing additional packages.
   r - reset all router configuration
 + l - load interface driver
 * a - configure ip address and gateway
   d - setup dhcp client
   s - setup dhcp server
   p - setup pppoe client
   t - setup pptp client
   x - exit menu
your choice [press Enter to configure ip address and gateway]: a
```

To configure IP address and gateway, press **a** or [Enter], if the **a** choice is marked with an asterisk symbol ('*').

```
 * a - add ip address
 - g - setup default gateway
   x - exit menu
your choice [press Enter to add ip address]: a
```

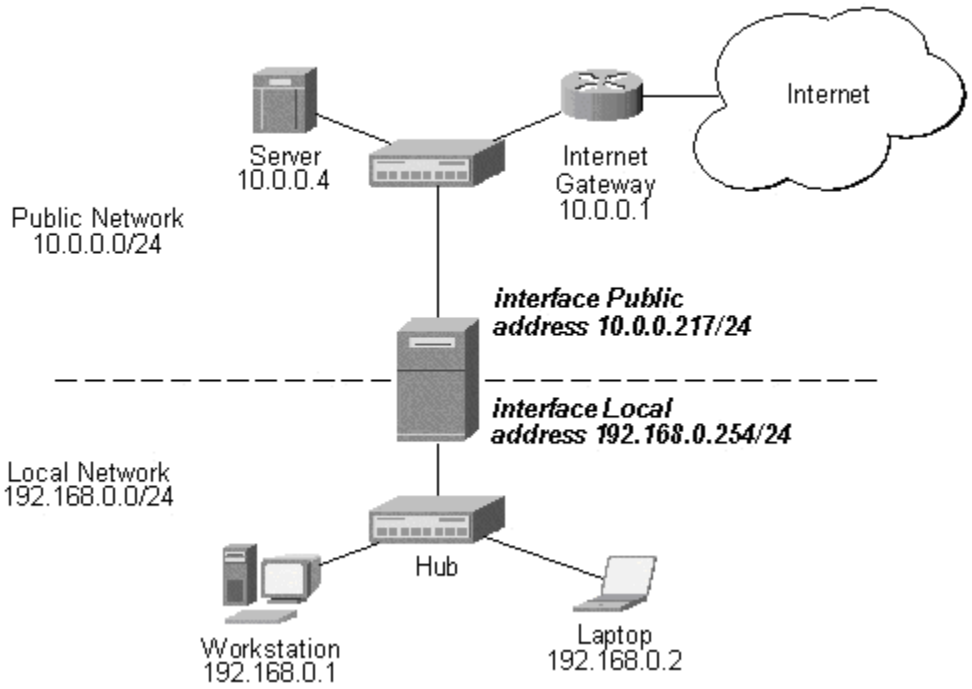
Choose **a** to add an IP address. At first, setup will ask you for an interface to which the address will be assigned. If the setup offers you an undesirable interface, erase this choice, and press the [Tab] key twice to see all available interfaces. After the interface is chosen, assign IP address and network mask on it:

```
your choice: a
enable interface:
ether1  ether2  wlan1
enable interface: ether1
ip address/netmask: 10.1.0.66/24
#Enabling interface
/interface enable ether1
#Adding IP address
/ip address add address=10.1.0.66/24 interface=ether1 comment="added by setup"
 + a - add ip address
 * g - setup default gateway
   x - exit menu
your choice: x
```

Basic Examples

Example

Assume you need to configure the HauteSpot router for the following network setup:



In the current example we use two networks:

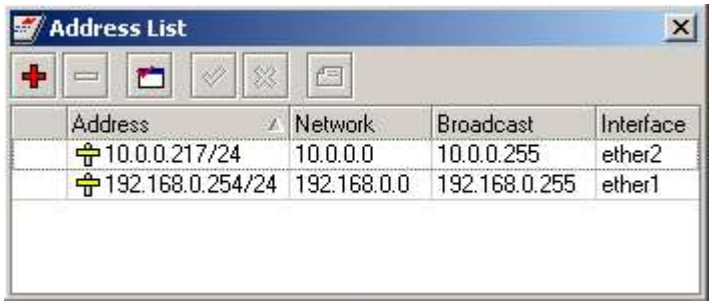
- The local LAN with network address 192.168.0.0 and 24-bit netmask: 255.255.255.0. The router's address is 192.168.0.254 in this network
- The ISP's network with address 10.0.0.0 and 24-bit netmask 255.255.255.0. The router's address is 10.0.0.217 in this network

The addresses can be added and viewed using the following commands:

```
[admin@HauteSpot] ip address> add address 10.0.0.217/24 interface Public
[admin@HauteSpot] ip address> add address 192.168.0.254/24 interface Local
[admin@HauteSpot] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
  #   ADDRESS            NETWORK         BROADCAST       INTERFACE
  0   10.0.0.217/24      10.0.0.217      10.0.0.255      Public
  1   192.168.0.254/24   192.168.0.0     192.168.0.255   Local
[admin@HauteSpot] ip address>
```

Here, the network mask has been specified in the value of the address argument. Alternatively, the argument 'netmask' could have been used with the value '255.255.255.0'. The network and broadcast addresses were not specified in the input since they could be calculated automatically.

Please **note** that the addresses assigned to different interfaces of the router should belong to different networks.



Viewing Routes

You can see two dynamic (D) and connected (C) routes, which have been added automatically when the addresses were added in the example above:

```
[admin@HauteSpot] ip route> print
Flags:  A - active, X - disabled, I - invalid, D - dynamic, C - connect,
       S - static, r - rip, b - bgp, o - ospf, d - dynamic
    #    DST-ADDRESS        G GATEWAY         DISTANCE INTERFACE
    0 ADC 192.168.0.0/24     r 0.0.0.0          0        Local
    1 ADC 10.0.0.0/24        r 0.0.0.0          0        Public
[admin@HauteSpot] ip route> print detail
Flags: A - active, X - disabled, I - invalid, D - dynamic, C - connect,
       S - static, r - rip, b - bgp, o - ospf, d - dynamic
 0 ADC dst-address=192.168.0.0/24 prefsrc=192.168.0.254 interface=Local
scope=10

 1 ADC dst-address=10.0.0.0/24 prefsrc=10.0.0.217 interface=Public scope=10

[admin@HauteSpot] ip route>
```

These routes show, that IP packets with destination to 10.0.0.0/24 would be sent through the interface Public, whereas IP packets with destination to 192.168.0.0/24 would be sent through the interface Local. However, you need to specify where the router should forward packets, which have destination other than networks connected directly to the router.

Adding Default Routes

In the following example the `default route` (destination 0.0.0.0 (any), netmask 0.0.0.0 (any)) will be added. In this case it is the ISP's gateway 10.0.0.1, which can be reached through the interface **Public**

```
[admin@HauteSpot] ip route> add gateway=10.0.0.1
[admin@HauteSpot] ip route> print
Flags: X - disabled, I - invalid, D - dynamic, J - rejected,
C - connect, S - static, R - rip, O - ospf, B - bgp
    #     DST-ADDRESS        G GATEWAY         DISTANCE INTERFACE
    0 ADC 192.168.0.0/24                                 Local
    1 ADC 10.0.0.0/24                                    Public
```

```
    2 A S 0.0.0.0/0          r 10.0.0.1         0      Public
[admin@HauteSpot] ip route>
```

Here, the default route is listed under #2. As we see, the gateway 10.0.0.1 can be reached through the interface 'Public'. If the gateway was specified incorrectly, the value for the argument 'interface' would be unknown.

Notes

You cannot add two routes to the same destination, i.e., destination-address/netmask! It applies to the default routes as well. Instead, you can enter multiple gateways for one destination. For more information on IP routes, please read the Routes, Equal Cost Multipath Routing, Policy Routing manual.

If you have added an unwanted static route accidentally, use the **remove** command to delete the unneeded one. You will not be able to delete dynamic (DC) routes. They are added automatically and represent routes to the networks the router connected directly.

Testing the Network Connectivity

From now on, the **/ping** command can be used to test the network connectivity on both interfaces. You can reach any host on both connected networks from the router.

How the **/ping** command works:

```
[admin@HauteSpot] ip route> /ping 10.0.0.4
10.0.0.4 64 byte ping: ttl=255 time=7 ms
10.0.0.4 64 byte ping: ttl=255 time=5 ms
10.0.0.4 64 byte ping: ttl=255 time=5 ms
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 5/5.6/7 ms
[admin@HauteSpot] ip route>
[admin@HauteSpot] ip route> /ping 192.168.0.1
192.168.0.1 64 byte ping: ttl=255 time=1 ms
192.168.0.1 64 byte ping: ttl=255 time=1 ms
192.168.0.1 64 byte ping: ttl=255 time=1 ms
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 1/1.0/1 ms
[admin@HauteSpot] ip route>
```

The workstation and the laptop can reach (ping) the router at its local address 192.168.0.254, If the router's address 192.168.0.254 is specified as the default gateway in the TCP/IP configuration of both the workstation and the laptop, then you should be able to ping the router:

```
C:\>ping 192.168.0.254
Reply from 192.168.0.254: bytes=32 time=10ms TTL=253
```

```
Reply from 192.168.0.254: bytes=32 time<10ms TTL=253
Reply from 192.168.0.254: bytes=32 time<10ms TTL=253

C:\>ping 10.0.0.217
Reply from 10.0.0.217: bytes=32 time=10ms TTL=253
Reply from 10.0.0.217: bytes=32 time<10ms TTL=253
Reply from 10.0.0.217: bytes=32 time<10ms TTL=253

C:\>ping 10.0.0.4
Request timed out.
Request timed out.
Request timed out.
```

Notes

You cannot access anything beyond the router (network 10.0.0.0/24 and the Internet), unless you do the one of the following:

- Use source network address translation (masquerading) on the HauteSpot router to 'hide' your private LAN 192.168.0.0/24 (see the information below), or
- Add a static route on the ISP's gateway 10.0.0.1, which specifies the host 10.0.0.217 as the gateway to network 192.168.0.0/24. Then all hosts on the ISP's network, including the server, will be able to communicate with the hosts on the LAN

To set up routing, it is required that you have some knowledge of configuring TCP/IP networks. We strongly recommend that you obtain more knowledge, if you have difficulties configuring your network setups.

Advanced Configuration Tasks

Description

Next will be discussed situation with 'hiding' the private LAN 192.168.0.0/24 'behind' one address 10.0.0.217 given to you by the ISP.

Application Example with Masquerading

If you want to 'hide' the private LAN 192.168.0.0/24 'behind' one address 10.0.0.217 given to you by the ISP, you should use the source network address translation (masquerading) feature of the HauteSpot router. Masquerading is useful, if you want to access the ISP's network and the Internet appearing as all requests coming from the host 10.0.0.217 of the ISP's network. The masquerading will change the source IP address and port of the packets originated from the network 192.168.0.0/24 to the address 10.0.0.217 of the router when the packet is routed through it.

Masquerading conserves the number of global IP addresses required and it lets the whole network use a single IP address in its communication with the world.

To use masquerading, a source NAT rule with action 'masquerade' should be added to the firewall configuration:

```
[admin@HauteSpot] ip firewall nat> add chain=srcnat action=masquerade out-
interface=Public
[admin@HauteSpot] ip firewall nat> print
Flags: X - disabled, I - invalid, D - dynamic
 0   chain=srcnat out-interface=Public action=masquerade
```

Notes

Please consult Network Address Translation for more information on masquerading.

Example with Bandwidth Management

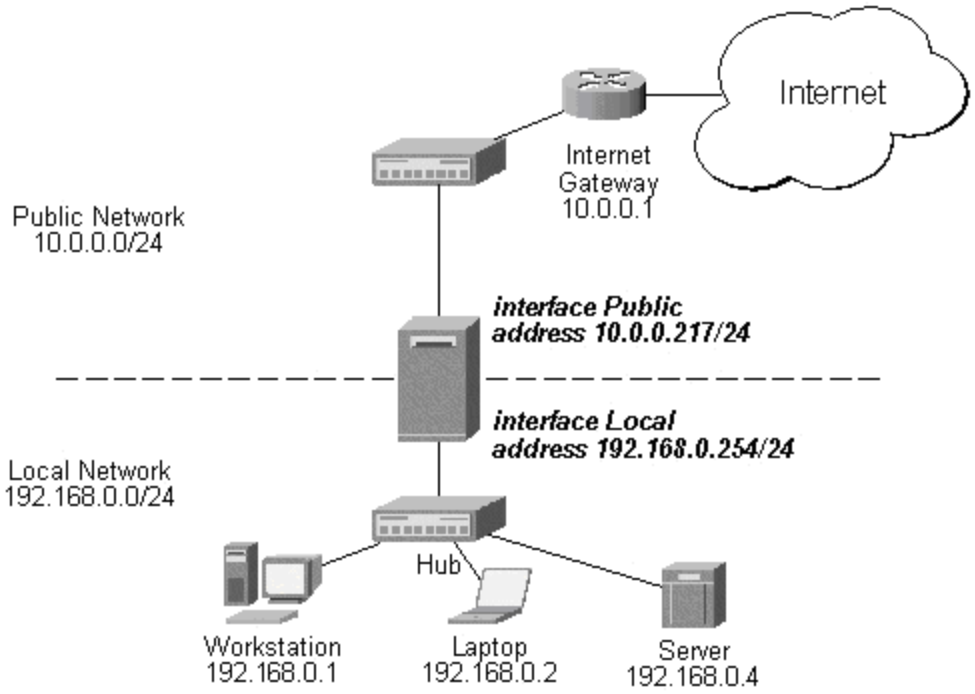
Assume you want to limit the bandwidth to 128kbps on downloads and 64kbps on uploads for all hosts on the LAN. Bandwidth limitation is done by applying queues for outgoing interfaces regarding the traffic flow. It is enough to add a single queue at the HauteSpot router:

```
[admin@HauteSpot] queue simple> add max-limit=64000/128000 interface=Local
[admin@HauteSpot] queue simple> print
Flags: X - disabled, I - invalid, D - dynamic
 0    name="queue1" target-address=0.0.0.0/0 dst-address=0.0.0.0/0
      interface=Local queue=default/default priority=8 limit-at=0/0
      max-limit=64000/128000 total-queue=default
[admin@HauteSpot] queue simple>
```

Leave all other parameters as set by default. The limit is approximately 128kbps going to the LAN (download) and 64kbps leaving the client's LAN (upload).

Example with NAT

Assume we have moved the server in our previous examples from the public network to our local one:



The server's address is now 192.168.0.4, and we are running web server on it that listens to the TCP port 80. We want to make it accessible from the Internet at address:port 10.0.0.217:80. This can be done by means of Static Network Address translation (NAT) at the HauteSpot Router. The Public address:port 10.0.0.217:80 will be translated to the Local address:port 192.168.0.4:80. One destination NAT rule is required for translating the destination address and port:

```
[admin@HauteSpot] ip firewall nat> add chain=dstnat action=dst-nat
protocol=tcp dst-address=10.0.0.217/32
                                dst-port=80 to-addresses=192.168.0.4
[admin@HauteSpot] ip firewall nat> pr
Flags: X - disabled, I - invalid, D - dynamic
 0   chain=dstnat dst-address=10.0.0.217/32 protocol=tcp dst-port=80
     action=dst-nat to-addresses=192.168.0.4 to-ports=0-65535
```

HauteRouterOS BGP Command Reference

Document revision: 1.5 (Tuesday, October 30, 2007)

Applies to: HauteRouterOS V3.0

General Information

Summary

The Border Gateway Protocol (BGP) allows setting up an interdomain dynamic routing system that automatically updates routing tables of devices running BGP in case of network topology changes.

HauteRouterOS supports BGP Version 4, as defined in RFC1771.

Starting from version v2.9 HauteRouterOS has a brand new BGP implementation, which provides advanced functionality not available in the previous versions.

Quick Setup Guide

To configure a BGP instance with AS number of **200** and establish a BGP session to the **10.0.11.11** peer from the AS **100**, redistributing connected and static routes only, you should do the following:

- Configure default BGP instance:

```
[admin@rb12] > /routing bgp instance set default as=200 redistribute-
static=yes redistribute-connected=yes
[admin@rb12] > /routing bgp instance print
Flags: X - disabled
 0   as=200 router-id=0.0.0.0 redistribute-static=yes redistribute-
connected=yes redistribute-rip=no
redistribute-ospf=no redistribute-other-bgp=no
     name="default" out-filter=""
```

```
[admin@rb12] >
```

- Add BGP peer:

```
[admin@rb12] > /routing bgp peer add remote-address=10.0.11.11 remote-
as=100 instance=default
[admin@rb12] > /routing bgp peer print
Flags: X - disabled
 0   remote-address=10.0.11.11 remote-as=100 multihop=no in-filter=""
out-filter="" keepalive-time=0s hold-time=0s ttl=1
```

```
[admin@rb12] >
```

Note, that the peer should be configured accordingly in order BGP to work.

Attention! In this scenario the router has no input or output filters configured. This means that it can redistribute lots of unnecessary or harmful information to its peers. Always consider configuring proper routing filters before you configure BGP peering.

Specifications

Packages required: ***routing-test***
License required: *Level3*
Submenu level: ***/routing bgp***
Standards and Technologies: RFC1771
Hardware usage: *requires additional RAM for storing routing information (128MB recommended)*

Description

The Border Gateway Protocol (BGP) is the core routing protocol of the Internet. It maintains a table of routes 'prefixes', which specify network layer reachability information (NLRI) between autonomous systems (AS). BGP is described as path vector protocol or policy routing protocol, referring to the way it chooses the best route towards destination. Unlike many other routing protocols, BGP does not use technical metrics to select the best path but rather administrative policies. The current version of BGP, Border Gateway Protocol 4, is specified in RFC 1771.

The routes learned by BGP protocol are installed in the route list with the distance of 200 for iBGP (Internal BGP) routes and of 20 for eBGP (External BGP) routes.

Additional Resources

- http://www.ietf.org/rfc/rfc1771.txt
- http://www.cisco.com/univercd/cc/td/doc/cisintwk/ics/icsbgp4.htm
- http://www.cisco.com/univercd/cc/td/doc/cisintwk/idg4/nd2003.htm

Instances

Description

Submenu level: ***/routing bgp instance***

Property Description

as (*integer*: 0..65535) - BGP autonomous system number
name (*name*; default: **""**) - BGP instance name
out-filter (*name*; default: **""**) - output routing filter used by this BGP instance
redistribute-connected (yes | no; default: **no**) - if enabled, the router will redistribute the information about all connected routes, i.e., routes to the networks that can be directly reached

redistribute-ospf (yes | no; default: **no**) - if enabled, the router will redistribute the information about all routes learned by the OSPF protocol
redistribute-other-bgp (yes | no; default: **no**) - specifies whether this BGP instance should redistribute to its peers routes learned by other BGP instances
redistribute-rip (yes | no; default: **no**) - if enabled, the router will redistribute the information about all routes learned by RIP protocol
redistribute-static (yes | no; default: **no**) - if enabled, the router will redistribute the information about all static routes added to its routing database, i.e., routes that have been created using the **/ip route add** command on the router
router-id (*IP address*; default: **0.0.0.0**) - the router identification string in form of an IP address. If no **router-id** is specified, it will be selected automatically based on the routing information

Peers

Submenu level: ***/routing bgp peer***

Description

You need to specify the BGP peer with whom you want to exchange the routing information. The BGP exchanges routing information only if it can establish a TCP connection to its peer. You can add as many peers as required.

Property Description

hold-time (*time*) - specifies the BGP Hold Time value to use when negotiating with peers. According to BGP specifications, if router does not receive successive KEEPALIVE and/or UPDATE and/or NOTIFICATION messages within the period specified in the Hold Time field of the OPEN message, then the BGP connection to the peer will be closed
in-filter (*name*; default: **""**) - name of the routing filter that is applied to incoming routing update messages
keepalive-time (*time*) - specifies the time interval between successive KEEPALIVE messages. BGP process will negotiate the keepalive time with the neighbour upon connection establishment
multihop (yes | no; default: **no**) - if enabled, allows BGP sessions, even when the neighbour is not on a directly connected segment. The multihop session is not established if the only route to the multi-hop peer's address is the default route (0.0.0.0/0)
out-filter (*name*; default: **""**) - name of the routing filter that is applied to outgoing routing update messages
remote-address (*IP address*; default: **0.0.0.0**) - address of the remote peer
remote-as (*integer*; default: **0**) - AS number of the remote peer

HauteRouterOS BGP Routing Filters

Document revision: 1.4 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

Border Gateway Protocol (BGP) Routing filters allow to alter attributes of the route for the NLRI prefixes or completely exclude particular NLRI prefixes with routes from the BGP routing update message.

Specifications

Packages required: ***routing***
License required: *Level3*
Submenu level: ***/routing filter***
Standards and Technologies: RFC1771
Hardware usage: *Not significant*

Description

BGP filtering refers to the ability of BGP peer to apply administrative policies to incoming and outgoing routing update messages. These policies are implemented as rules organized in chains. The following manual uses terms 'chain' and 'filter' interchengeably. Each rule consists of two parts, one of them specifies to which prefixes the rule applies to and the other tells the router what to do with these prefixes. A rule with no arguments applies to all prefixes and implies **accept** action.

The routing filters may be applied to incoming and outgoing routing update messages for a specific BGP peer and to outgoing BGP update messages for a particular BGP instance. Note, that in case both BGP instance and BGP peer outgoing filters are applied, BGP instance filters take precedence.

Additional Resources

- http://www.ietf.org/rfc/rfc1771.txt
- http://www.cisco.com/univercd/cc/td/doc/cisintwk/ics/icsbgp4.htm
- http://www.cisco.com/univercd/cc/td/doc/cisintwk/idg4/nd2003.htm

Filter Rules

Property Description

action (accept | discard | jump | none | reject | return; default: **none**) - action to perform on route or route attributes for the NLRI prefixes that match the rule
accept - accept the routing information for the matching NLRI prefix
discard - completely exclude matching prefix from the BGP processing. The route will be deleted from the incoming BGP routing update message, thus reducing memory usage on the router. For outgoing BGP update messages the **discard** action is equal to **reject**
jump - pass control to another filter list that should be specified as **jump-target** parameter
none - do not perform any action and pass execution to the next rule in chain. The **none** action is not displayed by **print** command
reject - reject the routing information for matching prefix. The prefix from incoming BGP routing update message is be shown with **R** (rejected) flag in the **/ip route print** command output. The prefix is suppressed from outgoing routing update message
return - return to the previous chain from which a jump to the current chain took place
as-path (*text*) - unanchored pattern to be searched inside AS_PATH attribute of the route. Optional **^** sign preceiding parameter value restricts match to the beginning of AS_PATH attribute, while **$** sign, which follows **as-path** value, restricts the match to the end of AS_PATH
as-path-length (*integer-integer*) - length of the AS_PATH attribute, representing the number of ASs that have been traversed. Note that multiple AS_SETs are combined together and counted as 1 AS
atomic-aggregate (absent | present) - match for the ATOMIC_AGGREGATE BGP attribute
chain (*text*) - chain name to place this rule in. If a chain with the specified name does not exist it will be automatically created
distance (*integer-integer*; default: **no**) - protocol-independent administrative distance used to compare routes obtained from different sources
jump-target (*name*) - name of the target chain to jump to, if the **action=jump** is used
local-pref (*integer-integer*) - match for the LOCAL_PREF BGP attribute
match-chain (*name*) - the name of the chain which is used to evaluate the route. If the chain accepts the route, **match-chain** property produces a true match
med (*integer-integer*) - match for the MULTI_EXIT_DISC BGP attribute
origin (igp | egp | incomplete) - match for the ORIGIN BGP attribute
prefix (*IP address/netmask* | *IP address-IP address*) - match for the NLRI prefix
prefix-length (*integer-integer*) - match for the NLRI prefix length
prefsrc (*IP address/netmask* | *IP address-IP address*) - match for the preferred source IP address of the route
route-comment (*text*) - match for the route comment
routing-mark (*text*) - match for the routing mark. A routing mark identifies certain routes for successive processing
scope (*integer*: 0..255-*integer*: 0..255) - **scope** and **target-scope** are used to recursively lookup next hop address for the route. Routes that are used to lookup the next hop address for a given route should have **scope** value equal or less then the **target-scope** value of this route
set-check-gateway (ping | arp) - specifies that the router should check whether the gateway for the particular route is reachable by using either ping or arp request prior to sending anything using this route

set-disabled - disables the route. Disabled routes are not considered by BGP best path selection algorithm
set-distance (*integer*: 0..255) - sets administrative distance for a route. The distance is protocol-independent and is used to compare routes obtained from different sources
set-localpref (*integer*: 0..4294967295) - specifies LOCAL_PREF BGP attribute value for the route
set-med (*integer*: 0..4294967295) - sets MULTI_EXIT_DISC BGP attribute
set-nexthop (*IP address*) - sets next hop IP address for the route
set-prefsrc (*IP address*) - sets preffered source address for the route
set-prepend (*integer*: 0..16) - specifies how many times the router should prepend its AS number to the AS_PATH BGP attribute value for this route
set-route-comment (*text*) - specifies comment for the route
set-routing-mark (*text*) - sets routing mark for the route
set-scope (*integer*: 0..255) - sets scope for the route. **Scope** and **target-scope** are used to recursively lookup next hop address for the route. Routes that are used to lookup the next hop address for a given route should have **scope** value equal or less then the **target-scope** value of this route
set-target-scope (*integer*: 0..255) - sets target scope for the route. **Scope** and **target-scope** are used to recursively lookup next hop address for the route. Routes that are used to lookup the next hop address for a given route should have **scope** value equal or less then the **target-scope** value of this route
set-weight (*integer*: -2147483648..2147483647) - specifies weight for the route. Route weight is used by BGP best path selection algoritm to select the best route towards destination
target-scope (*integer*: 0..255-*integer*: 0..255) - **scope** and **target-scope** are used to recursively lookup next hop address for the route. Routes that are used to lookup the next hop address for a given route should have **scope** value equal or less then the **target-scope** value of this route
type (absent | present) - match for the ATOMIC_AGGREGATE BGP attribute
unset (*multiple choice:* prefsrc | routing-mark | check-gateway | disabled) - unsets specified parameters of the route
weight (*integer*: -2147483648..2147483647) - match for the weight of the route

HauteRouterOS Bridge

Document revision: 2.1 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

MAC level bridging of Ethernet, Ethernet over IP (EoIP), Prism, Atheros and RadioLAN interfaces are supported. All 802.11a, 802.11b, and 802.11g **client** wireless interfaces (**ad-hoc**, **infrastructure** or **station** mode) do not support this because of the limitations of 802.11. However, it is possible to bridge over the Prism

and Atheros based links using the WDS feature (for Atheros and Prism chipset based cards) or Ethernet over IP protocol.

For preventing loops in a network, you can use the Spanning Tree Protocol (STP). This protocol is also used for configurations with backup links.

Main features:

- Spanning Tree Protocol (STP)
- Multiple bridge interfaces
- Bridge associations on a per-interface basis
- MAC address table can be monitored in real time
- IP address assignment for router access
- Bridge interfaces can be filtered and NATed
- Support for brouting based on bridge packet filter

Quick Setup Guide

To put interface **ether1** and **ether2** in a bridge.

1. Add a bridge interface, called **MyBridge**:

```
/interface bridge add name="MyBridge" disabled=no
```

2. Add **ether1** and **ether2** to **MyBridge** interface:

```
/interface bridge port set ether1,ether2 bridge=MyBridge
```

Specifications

Packages required: ***system***
License required: *Level3*
Submenu level: ***/interface bridge***
Standards and Technologies: IEEE801.1D
Hardware usage: *Not significant*

Description

Ethernet-like networks (Ethernet, Ethernet over IP, IEEE802.11 in ap-bridge or bridge mode, WDS, VLAN) can be connected together using MAC bridges. The bridge feature allows the interconnection of hosts connected to separate LANs (using EoIP, geographically distributed networks can be bridged as well if any kind of IP network interconnection exists between them) as if they were attached to a single LAN. As bridges are transparent, they do not appear in traceroute list, and no utility can make a distinction between a host working in one LAN and a host working in another

LAN if these LANs are bridged (depending on the way the LANs are interconnected, latency and data rate between hosts may vary).

Network loops may emerge (intentionally or not) in complex topologies. Without any special treatment, loops would prevent network from functioning normally, as they would lead to avalanche-like packet multiplication. Each bridge runs an algorithm which calculates how the loop can be prevented. STP allows bridges to communicate with each other, so they can negotiate a loop free topology. All other alternative connections that would otherwise form loops, are put to standby, so that should the main connection fail, another connection could take its place. This algorithm exchange configuration messages (BPDU - Bridge Protocol Data Unit) periodically, so that all bridges would be updated with the newest information about changes in network topology. STP selects root bridge which is responosible for network reconfiguration, such as blocking and opening ports of the other bridges. The root bridge is the bridge with lowest bridge ID.

Additional Resources

http://ebtables.sourceforge.net/

Bridge Interface Setup

Submenu level: ***/interface bridge***

Description

To combine a number of networks into one bridge, a bridge interface should be created (later, all the desired interfaces should be set up as its ports). One MAC address will be assigned to all the bridged interfaces (the smallest MAC address will be chosen automatically).

Property Description

ageing-time (*time*; default: **5m**) - how long a host information will be kept in the bridge database
arp (disabled | enabled | proxy-arp | reply-only; default: **enabled**) - Address Resolution Protocol setting
forward-delay (*time*; default: **15s**) - time which is spent during the initialization phase of the bridge interface (i.e., after router startup or enabling the interface) in listening/learning state before the bridge will start functioning normally
garbage-collection-interval (*time*; default: **4s**) - how often to drop old (expired) host entries in the bridge database. The garbage collection process expurges the entries older than defined by the **ageing-time** property
hello-time (*time*; default: **2s**) - how often send hello packets to other bridges
mac-address (*read-only: MAC address*) - MAC address for the interface
max-message-age (*time*; default: **20s**) - how long to remember Hello messages received from other bridges

mtu (*integer*; default: **1500**) - Maximum Transmission Unit
name (*name*; default: **bridgeN**) - a descriptive name of the bridge interface
priority (*integer*: 0..65535; default: **32768**) - bridge interface priority. The priority argument is used by Spanning Tree Protocol to determine, which port remains enabled if at least two ports form a loop
stp (no | yes; default: **no**) - whether to enable the Spanning Tree Protocol. Bridging loops will only be prevented if this property is turned on

Example

To add and enable a bridge interface that will forward all the protocols:

```
[admin@Hautespot] interface bridge> add; print
Flags: X - disabled, R - running
 0  R name="bridge1" mtu=1500 arp=enabled mac-address=61:64:64:72:65:73 stp=no
      priority=32768 ageing-time=5m forward-delay=15s
      garbage-collection-interval=4s hello-time=2s max-message-age=20s
[admin@Hautespot] interface bridge> enable 0
```

Port Settings

Submenu level: ***/interface bridge port***

Description

The submenu is used to enslave interfaces in a particular bridge interface.

Property Description

bridge (*name*; default: **none**) - the bridge interface the respective interface is grouped in
none - the interface is not grouped in any bridge
interface (*read-only: name*) - interface name, which is to be included in a bridge
path-cost (*integer*: 0..65535; default: **10**) - path cost to the interface, used by STP to determine the 'best' path
priority (*integer*: 0..255; default: **128**) - interface priority compared to other interfaces, which are destined to the same network

Notes

Starting from version 2.9.9, the ports in this lists should be added, not set, see the following examples.

Example

To group **ether1** and **ether2** in the already created **bridge1** bridge (versions before 2.9.9):

```
[admin@Hautespot] interface bridge port> set ether1,ether2 bridge=bridge1
[admin@Hautespot] interface bridge port> print
 # INTERFACE   BRIDGE PRIORITY PATH-COST
 0 ether1      bridge1   128      10
 1 ether2      bridge1   128      10
 2 wlan1       none      128      10
[admin@Hautespot] interface bridge port>
```

To group **ether1** and **ether2** in the already created **bridge1** bridge (versions from 2.9.9):

```
[admin@Hautespot] interface bridge port> add ether1,ether2 bridge=bridge1
[admin@Hautespot] interface bridge port> print
 # INTERFACE   BRIDGE PRIORITY PATH-COST
 0 ether1      bridge1   128      10
 1 ether2      bridge1   128      10
[admin@Hautespot] interface bridge port>
```

Note that there is no **wlan1** interface anymore, as it is not added as bridge port.

Bridge Monitoring

Command name: ***/interface bridge monitor***

Description

Used to monitor the current status of a bridge.

Property Description

bridge-id (*text*) - the bridge ID, which is in form of bridge-priority.bridge-MAC-address
designated-root (*text*) - ID of the root bridge
path-cost (*integer*) - the total cost of the path to the root-bridge
root-port (*name*) - port to which the root bridge is connected to

Example

To monitor a bridge:

```
[admin@Hautespot] interface bridge> monitor bridge1
          bridge-id: 32768.00:02:6F:01:CE:31
    designated-root: 32768.00:02:6F:01:CE:31
```

```
          root-port: ether2
          path-cost: 180

[admin@Hautespot] interface bridge>
```

Bridge Port Monitoring

Command name: ***/interface bridge port monitor***

Description

Statistics of an interface that belongs to a bridge

Property Description

designated-port (*text*) - port of designated-root bridge
designated-root (*text*) - ID of bridge, which is nearest to the root-bridge
port-id (*integer*) - port ID, which represents from port priority and port number, and is unique
status (disabled | blocking | listening | learning | forwarding) - the status of the bridge port:
disabled - the interface is disabled. No frames are forwarded, no Bridge Protocol Data Units (BPDUs) are heard
blocking - the port does not forward any frames, but listens for BPDUs
listening - the port does not forward any frames, but listens to them
learning - the port does not forward any frames, but learns the MAC addresses
forwarding - the port forwards frames, and learns MAC addresses

Example

To monitor a bridge port:

```
[admin@Hautespot] interface bridge port> mo 0
               status: forwarding
              port-id: 28417
      designated-root: 32768.00:02:6F:01:CE:31
    designated-bridge: 32768.00:02:6F:01:CE:31
      designated-port: 28417
      designated-cost: 0
-- [Q quit|D dump|C-z pause]
```

Bridge Host Monitoring

Command name: ***/interface bridge host***

Property Description

age (*read-only: time*) - the time since the last packet was received from the host
bridge (*read-only: name*) - the bridge the entry belongs to
local (*read-only: flag*) - whether the host entry is of the bridge itself (that way all local interfaces are shown)
mac-address (*read-only: MAC address*) - host's MAC address
on-interface (*read-only: name*) - which of the bridged interfaces the host is connected to

Example

To get the active host table:

```
[admin@Hautespot] interface bridge host> print
Flags: L - local
   BRIDGE              MAC-ADDRESS       ON-INTERFACE       AGE
   bridge1             00:00:B4:5B:A6:58 ether1             4m48s
   bridge1             00:30:4F:18:58:17 ether1             4m50s
 L bridge1             00:50:08:00:00:F5 ether1             0s
 L bridge1             00:50:08:00:00:F6 ether2             0s
   bridge1             00:60:52:0B:B4:81 ether1             4m50s
   bridge1             00:C0:DF:07:5E:E6 ether1             4m46s
   bridge1             00:E0:C5:6E:23:25 prism1             4m48s
   bridge1             00:E0:F7:7F:0A:B8 ether1             1s
[admin@Hautespot] interface bridge host>
```

Bridge Firewall General Description

Specifications

Submenu level: ***/interface bridge filter***, ***/interface bridge nat***, ***/interface bridge broute***

Description

The bridge firewall implements packet filtering and thereby provides security functions that are used to manage data flow to, from and through bridge

Note that packets between bridged interfaces, just like any other IP traffic, are also passed through the 'generic' **/ip firewall** rules (but bridging filters are always applied before IP filters/NAT of the built-in chain of the same name, except for the **output** which is executed after IP Firewall Output). These rules can be used with real, physical receiving/transmitting interfaces, as well as with bridge interface that simply groups the bridged interfaces.

There are three bridge filter tables:

- **filter** - bridge firewall with three predefined chains:

    - **input** - filters packets, which destination is the bridge (including those packets that will be routed, as they are anyway destined to the bridge MAC address)
    - **output** - filters packets, which come from the bridge (including those packets that has been routed normally)
    - **forward** - filters packets, which are to be bridged (note: this chain is not applied to the packets that should be routed through the router, just to those that are traversing between the ports of the same bridge)
- **nat** - bridge network address translation provides ways for changing source/destination MAC addresses of the packets traversing a bridge. Has two built-in chains:
    - **scnat** - used for "hiding" a host or a network behind a different MAC address. This chain is applied to the packets leaving the router through a bridged interface
    - **dstnat** - used for redirecting some pakets to another destinations
- **broute** - makes bridge a brouter - router that performs routing on some of the packets, and bridging - on others. Has one predefined chain: **brouting**, which is traversed right after a packet enters an enslaved interface (before "Bridging Decision")

Note: the bridge destination NAT is executed before bridging desision

You can put packet marks in bridge firewall (filter, broute and NAT), which are the same as the packet marks in IP firewall put by mangle. So packet marks put by bridge firewall can be used in IP firewall, and vice versa

General bridge firewall properties are described in this section. Some parameters that differ between nat, broute and filter rules are described in further sections.

Property Description

802.3-sap (*integer*) - DSAP (Destination Service Access Point) and SSAP (Source Service Access Point) are 2 one byte fields, which identify the network protocol entities which use the link layer service. These bytes are always equal. Two hexadecimal digits may be specified here to match an SAP byte
802.3-type (*integer*) - Ethernet protocol type, placed after the IEEE 802.2 frame header. Works only if **802.3-sap** is 0xAA (SNAP - Sub-Network Attachment Point header). For example, AppleTalk can be indicated by SAP code of 0xAA followed by a SNAP type code of 0x809B
arp-dst-address (*IP address*; default: **0.0.0.0/0**) - ARP destination address
arp-dst-mac-address (*MAC address*; default: **00:00:00:00:00:00**) - ARP destination MAC address
arp-hardware-type (*integer*; default: **1**) - ARP hardware type. This normally Ethernet (Type 1)
arp-opcode (arp-nak | drarp-error | drarp-reply | drarp-request | inarp-request | reply | reply-reverse | request | request-reverse) - ARP opcode (packet type)
arp-nak - negative ARP reply (rarely used, mostly in ATM networks)
drarp-error - Dynamic RARP error code, saying that an IP address for the given MAC address can not be allocated
drarp-reply - Dynamic RARP reply, with a temporaty IP address assignment for a host

drarp-request - Dynamic RARP request to assign a temporary IP address for the given MAC address
inarp-request -
reply - standard ARP reply with a MAC address
reply-reverse - reverse ARP (RARP) reply with an IP address assigned
request - standard ARP request to a known IP address to find out unknown MAC address
request-reverse - reverse ARP (RARP) request to a known MAC address to find out unknown IP address (intended to be used by hosts to find out their own IP address, similarly to DHCP service)
arp-packet-type (*integer*) -
arp-src-address (*IP address*; default: **0.0.0.0/0**) - ARP source IP address
arp-src-mac-address (*MAC address*; default: **00:00:00:00:00:00**) - ARP source MAC address
chain (*text*) - bridge firewall chain, which the filter is functioning in (either a built-in one, or a user defined)
dst-address (*IP address*; default: **0.0.0.0/0**) - destination IP address (only if MAC protocol is set to IPv4)
dst-mac-address (*MAC address*; default: **00:00:00:00:00:00**) - destination MAC address
dst-port (*integer*: 0..65535) - destination port number or range (only for TCP or UDP protocols)
flow (*text*) - individual packet mark to match
in-bridge (*name*) - bridge interface through which the packet is coming in
in-interface (*name*) - physical interface (i.e., bridge port) through which the packet is coming in
ip-protocol (ipsec-ah | ipsec-esp | ddp | egp | ggp | gre | hmp | idpr-cmtp | icmp | igmp | ipencap | encap | ipip | iso-tp4 | ospf | pup | rspf | rdp | st | tcp | udp | vmtp | xns-idp | xtp) - IP protocol (only if MAC protocol is set to IPv4)
ipsec-ah - IPsec AH protocol
ipsec-esp - IPsec ESP protocol
ddp - datagram delivery protocol
egp - exterior gateway protocol
ggp - gateway-gateway protocol
gre - general routing encapsulation
hmp - host monitoring protocol
idpr-cmtp - idpr control message transport
icmp - internet control message protocol
igmp - internet group management protocol
ipencap - ip encapsulated in ip
encap - ip encapsulation
ipip - ip encapsulation
iso-tp4 - iso transport protocol class 4
ospf - open shortest path first
pup - parc universal packet protocol
rspf - radio shortest path first
rdp - reliable datagram protocol
st - st datagram mode
tcp - transmission control protocol
udp - user datagram protocol
vmtp - versatile message transport
xns-idp - xerox ns idp
xtp - xpress transfer protocol
jump-target (*name*) - if **action=jump** specified, then specifies the user-defined firewall chain to process the packet
limit (*integer*/*time*{0,1},*integer*) - restricts packet match rate to a given limit. Usefull to reduce the amount of log messages

Count - maximum average packet rate, measured in packets per second (pps), unless followed by **Time** option
Time - specifies the time interval over which the packet rate is measured
Burst - number of packets to match in a burst
log-prefix (*text*) - defines the prefix to be printed before the logging information
mac-protocol (*integer* | 802.2 | arp | ip | ipv6 | ipx | rarp | vlan) - Ethernet payload type (MAC-level protocol)
mark-flow (*name*) - marks existing flow
packet-type (broadcast | host | multicast | other-host) - MAC frame type:
broadcast - broadcast MAC packet
host - packet is destined to the bridge itself
multicast - multicast MAC packet
other-host - packet is destined to some other unicast address, not to the bridge itself
src-address (*IP address*; default: **0.0.0.0/0**) - source IP address (only if MAC protocol is set to IPv4)
src-mac-address (*MAC address*; default: **00:00:00:00:00:00**) - source MAC address
src-port (*integer*: 0..65535) - source port number or range (only for TCP or UDP protocols)
stp-flags (topology-change | topology-change-ack) - The BPDU (Bridge Protocol Data Unit) flags. Bridge exchange configuration messages named BPDU peridiocally for preventing from loop
topology-change - topology change flag is set when a bridge detects port state change, to force all other bridges to drop their host tables and recalculate network topology
topology-change-ack - topology change acknowledgement flag is sen in replies to the notification packets
stp-forward-delay (*time*: 0..65535) - forward delay timer
stp-hello-time (*time*: 0..65535) - stp hello packets time
stp-max-age (*time*: 0..65535) - maximal STP message age
stp-msg-age (*time*: 0..65535) - STP message age
stp-port (*integer*: 0..65535) - stp port identifier
stp-root-address (*MAC address*) - root bridge MAC address
stp-root-cost (*integer*: 0..65535) - root bridge cost
stp-root-priority (*time*: 0..65535) - root bridge priority
stp-sender-address (*MAC address*) - stp message sender MAC address
stp-sender-priority (*integer*: 0..65535) - sender priority
stp-type (config | tcn) - the BPDU type
config - configuration BPDU
tcn - topology change notification
vlan-encap (802.2 | arp | ip | ipv6 | ipx | rarp | vlan) - the MAC protocol type encapsulated in the VLAN frame
vlan-id (*integer*: 0..4095) - VLAN identifier field
vlan-priority (*integer*: 0..7) - the user priority field

Notes

stpmatchers are only valid if destination MAC address is 01:80:C2:00:00:00/FF:FF:FF:FF:FF:FF (Bridge Group address), also **stp** should be enabled.

ARP matchers are only valid if **mac-protocol** is **arp** or **rarp**

VLAN matchers are only valid for **vlan** ethernet protocol

IP-related matchers are only valid if **mac-protocol** is set as **ipv4**

802.3 matchers are only consulted if the actual frame is compliant with IEEE 802.2 and IEEE 802.3 standards (**note**: it is not the industry-standard Ethernet frame format used in most networks worldwide!). These matchers are ignored for other packets.

Bridge Packet Filter

Submenu level: ***/interface bridge filter***

Description

This section describes bridge packet filter specific filtering options, which were omitted in the general firewall description

Property Description

action (accept | drop | jump | log | mark | passthrough | return; default: **accept**) - action to undertake if the packet matches the rule, one of the:
accept - accept the packet. No action, i.e., the packet is passed through without undertaking any action, and no more rules are processed in the relevant list/chain
drop - silently drop the packet (without sending the ICMP reject message)
jump - jump to the chain specified by the value of the jump-target argument
log - log the packet
mark - mark the packet to use the mark later
passthrough - ignore this rule and go on to the next one. Acts the same way as a disabled rule, except for ability to count packets
return - return to the previous chain, from where the jump took place
out-bridge (*name*) - outgoing bridge interface
out-interface (*name*) - interface via packet is leaving the bridge

Bridge NAT

Submenu level: ***/interface bridge nat***

Description

This section describes bridge NAT options, which were omitted in the general firewall description

Property Description

action (accept | arp-reply | drop | dst-nat | jump | log | mark | passthrough | redirect | return | src-nat; default: **accept**) - action to undertake if the packet matches the rule, one of the:
accept - accept the packet. No action, i.e., the packet is passed through without undertaking any action, and no more rules are processed in the relevant list/chain
arp-reply - send a reply to an ARP request (any other packets will be ignored by this rule) with the specified MAC address (only valid in **dstnat** chain)
drop - silently drop the packet (without sending the ICMP reject message)
dst-nat - change destination MAC address of a packet (only valid in **dstnat** chain)
jump - jump to the chain specified by the value of the jump-target argument
log - log the packet
mark - mark the packet to use the mark later
passthrough - ignore this rule and go on to the next one. Acts the same way as a disabled rule, except for ability to count packets
redirect - redirect the packet to the bridge itself (only valid in **dstnat** chain)
return - return to the previous chain, from where the jump took place
src-nat - change source MAC address of a packet (only valid in **srcnat** chain)
out-bridge (*name*) - outgoing bridge interface
out-interface (*name*) - interface via packet is leaving the bridge
to-arp-reply-mac-address (*MAC address*) - source MAC address to put in Ethernet frame and ARP payload, when **action=arp-reply** is selected
to-dst-mac-address (*MAC address*) - destination MAC address to put in Ethernet frames, when **action=dst-nat** is selected
to-src-mac-address (*MAC address*) - source MAC address to put in Ethernet frames, when **action=src-nat** is selected

Bridge Brouting Facility

Submenu level: ***/interface bridge broute***

Description

This section describes broute facility specific options, which were omitted in the general firewall description

The Brouting table is applied to every packet entering a forwarding enslaved interface (i.e., it does not work on regular interfaces, which are not included in a bridge)

Property Description

action (accept | drop | dst-nat | jump | log | mark | passthrough | redirect | return; default: **accept**) - action to undertake if the packet matches the rule, one of the:
accept - let the bridging code decide, what to do with this packet
drop - extract the packet from bridging code, making it appear just like it would come from a not-bridged interface (no further bridge decisions or filters will be applied to this packet except if the packet would be router out to a bridged interface, in which case the packet would be processed normally, just like any other routed packet)
dst-nat - change destination MAC address of a packet (only valid in **dstnat** chain), an let bridging code to decide further actions
jump - jump to the chain specified by the value of the jump-target argument

log - log the packet
mark - mark the packet to use the mark later
passthrough - ignore this rule and go on to the next one. Acts the same way as a disabled rule, except for ability to count packets
redirect - redirect the packet to the bridge itself (only valid in **dstnat** chain), an let bridging code to decide further actions
return - return to the previous chain, from where the jump took place
to-dst-mac-address (*MAC address*) - destination MAC address to put in Ethernet frames, when **action=dst-nat** is selected

Troubleshooting

Description

- **Router shows that my rule is invalid**
  - in-interface, in-bridge (or in-bridge-port) is specified, but such an interface does not exist
  - there is an action=mark-packet, but no new-packet-mark
  - there is an action=mark-connection, but no new-connection-mark
  - there is an action=mark-routing, but no new-routing-mark

HauteRouterOS Certificate Management

Document revision: 2.3 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

SSL (Secure Socket Layer) is a security technology to ensure encrypted transactions over a public network. To protect the data, an encryption key should be negotiated. SSL protocol is using Certificates to negotiate a key for data encryption.

Specifications

Packages required: ***system***
License required: *Level1*
Submenu level: ***/certificate***
Standards and Technologies: SSLv2, SSLv3, TLS
Hardware usage: *high CPU usage*

Description

SSL technology was first introduced by Netscape to ensure secure transactions between browsers and web servers. When a browser requests a secure web page (usually on TCP port 443), a web server first sends a Certificate, which contains a public key for the encryption key negotiation to take place. After the encryption key is negotiated, the web server will send the requested page encrypted using this key to the browser (and also the browser will be able to submit its data securely to the server)

SSL Certificate confirms the web server identity. The Certificate contains information about its holder (like DNS name and Country), issuer (the entity has signed the Certificate) and also the public key used to negotiate the encryption key. In order a Certificate to play its role, it should be signed by a third party (Certificate Authority) which both parties trust. Modern browsers that support SSL protocol have a list of the Certificate Authorities they trust (the most known and trusted CA is VeriSign, but that is not the only one)

To use a Certificate (which contain a public key), server needs a private key. One of the keys is used for encryption, and the other - for decryption. It is important to understand, that both keys can encrypt and decrypt, but what is encrypted by one of them can be decrypted **only** by the another. Private key must be kept securely, so that nobody else can get it and use this certificate. Usually private key is encrypted with a passphrase.

Most trusted Certificate Authorities sell the service of signing Certificates (Certificates also have a finite validity term, so you will have to pay regularly). It is also possible to create a self-signed Certificate (you can create one on most UNIX/Linux boxes using openssl toolkit; all Root Certificate Authorities have self-signed Certificates), but if it is not present in a browser's database, the browser will pop up a security warning, saying that the Certificate is not trusted (note also that most browsers support importing custom Certificates to their databases).

Certificates

Submenu level: ***/certificate***

Description

Hautespot RouterOS can import Certificates for the SSL services it provides (only HotSpot for now). This submenu is used to manage Certificates for this services.

Property Description

name (*name*) - reference name
subject (*read-only: text*) - holder (subject) of the certificate
issuer (*read-only: text*) - issuer of the certificate

serial-number (*read-only: text*) - serial number of the certificate
invalid-before (*read-only: date*) - date the certificate is valid from
invalid-after (*read-only: date*) - date the certificate is valid until
ca (yes | no; default: **yes**) - whether the certificate is used for building or verifying certificate chains (as Certificate Authority)

Command Description

import - install new certificates
- **file-name** - import only this file (all files are searched for certificates by default)
- **passphrase** - passphrase for the found encrypted private key
- **certificates-imported** - how many new certificates were successfully imported
- **private-keys-imported** - how many private keys for existing certificates were successfully imported
- **files-imported** - how many files contained at least one item that was successfully imported
- **decryption-failures** - how many files could not be decrypted
- **keys-with-no-certificate** - how many public keys were successfully decrypted, but did not have matching certificate already installed

reset-certificate-cache - delete all cached decrypted public keys and rebuild the certificate cache
decrypt - decrypt and cache public keys
- **passphrase** - passphrase for the found encrypted private key
- **keys-decrypted** - how many keys were successfully decrypted and cached

create-certificate-request - creates an RSA certificate request to be signed by a Certificate Authority. After this, download both private key and certificate request files from the router. When you receive your signed certificate from the CA, upload it and the private key (that is made by this command) to a router and use **/certificate import** command to install it
- **certificate request file name** - name for the certificate request file (if it already exists, it will be overwritten). This is the original certificate that will be signed by the Certificate Authority
- **file name** - name of private key file. If such file does not exist, it will be created during the next step. Private key is used to encrypt the certificate
- **passphrase** - the passphrase that will be used to encrypt generated private key file. You must enter it twice to be sure you have not made any typing errors
- **rsa key bits** - number of bits for RSA (encryption) key. Longer keys take more time to generate. 4096 bit key takes about 30 seconds on Celeron 800 system to generate
- **country name** - (C) ISO two-character country code (e.g., **LV** for Latvia)
- **state or province name** - (ST) full name of state or province
- **locality name** - (L) locality (e.g. city) name
- **organization name** - (O) name of the organization or company
- **organization unit name** - (OU) organization unit name
- **common name** - (CN) the server's common name. For SSL web servers this must be the fully qualified domain name (FQDN) of the server that will use this certificate (like www.example.com). This is checked by web browsers
- **email address** - (Email) e-mail address of the person responsible for the certificate
- **challenge password** - the challenge password. It's use depends on your CA. It may be used to revoke this certificate
- **unstructured address** - unstructured address (like street address). Enter only if your CA accepts or requires it

Notes

Server certificates may have **ca** property set to **no**, but Certificate Authority certificates must have it set to **yes**

Certificates and encrypted private keys are imported from and exported to the router's FTP server. Public keys are not stored on a router in unencrypted form. Cached decrypted private keys are stored in encrypted form, using key that is derived from the router ID. Passphrases are not stored on router.

Configuration backup does not include cached decrypted private keys. After restoring backup all certificates with private keys must be decrypted again, using **decrypt** command with the correct passphrase.

No other certificate operations are possible while generating a key.

When making a certificate request, you may leave some of the fields empty. CA may reject your certificate request if some of these values are incorrect or missing, so please check what are the requirements of your CA

Example

To import a certificate and the respective private key already uploaded on the router:

```
[admin@Hautespot] certificate> import
passphrase: xxxx
       certificates-imported: 1
       private-keys-imported: 1
              files-imported: 2
         decryption-failures: 0
    keys-with-no-certificate: 1
[admin@Hautespot] certificate> print
Flags: K - decrypted-private-key, Q - private-key, R - rsa, D - dsa
  0 QR name="cert1" subject=C=LV,ST=.,O=.,CN=cert.test.mt.lv
       issuer=C=LV,ST=.,O=.,CN=third serial-number="01"
       invalid-before=sep/17/2003 11:56:19 invalid-after=sep/16/2004 11:56:19
       ca=yes

[admin@Hautespot] certificate> decrypt
passphrase: xxxx
    keys-decrypted: 1
[admin@Hautespot] certificate> print
Flags: K - decrypted-private-key, Q - private-key, R - rsa, D - dsa
  0 KR name="cert1" subject=C=LV,ST=.,O=.,CN=cert.test.mt.lv
       issuer=C=LV,ST=.,O=.,CN=third serial-number="01"
       invalid-before=sep/17/2003 11:56:19 invalid-after=sep/16/2004 11:56:19
       ca=yes

[admin@Hautespot] certificate>
```

Now the certificate may be used by HotSpot servlet:

```
[admin@Hautespot] ip service> print
Flags: X - disabled, I - invalid
```

```
  #   NAME                                    PORT  ADDRESS
CERTIFICATE
  0   telnet                                  23    0.0.0.0/0
  1   ftp                                     21    0.0.0.0/0
  2   www                                     8081  0.0.0.0/0
  3   hotspot                                 80    0.0.0.0/0
  4   ssh                                     22    0.0.0.0/0
  5   hotspot-ssl                             443   0.0.0.0/0          none

[admin@Hautespot] ip service> set hotspot-ssl certificate=
cert1  none
[admin@Hautespot] ip service> set hotspot-ssl certificate=cert1
[admin@Hautespot] ip service> print
Flags: X - disabled, I - invalid
  #   NAME                                    PORT  ADDRESS
CERTIFICATE
  0   telnet                                  23    0.0.0.0/0
  1   ftp                                     21    0.0.0.0/0
  2   www                                     8081  0.0.0.0/0
  3   hotspot                                 80    0.0.0.0/0
  4   ssh                                     22    0.0.0.0/0
  5   hotspot-ssl                             443   0.0.0.0/0          cert1

[admin@Hautespot] ip service>
```

HauteRouterOS Configuration Management

Document revision: 1.6 (Tuesday, October 30, 2007)

Applies to: HauteRouterOS V3.0

General Information

Summary

This manual introduces you with commands which are used to perform the following functions:

- system backup
- system restore from a backup
- configuration export
- configuration import
- system configuration reset

Description

The configuration backup can be used for backing up HauteRouterOS configuration to a binary file, which can be stored on the router or downloaded from it using FTP. The configuration restore can be used for restoring the router's configuration from a backup file.

The configuration export can be used for dumping out HauteRouterOS configuration to the console screen or to a text (script) file, which can be downloaded from the router using FTP. The configuration import can be used to import the router configuration script from a text file.

System reset command is used to erase all configuration on the router. Before doing that, it might be useful to backup the router's configuration.

Note! In order to be sure that the backup will not fail, **system backup load** command must be used on the same computer with the same hardware where **system backup save** was done.

System Backup

Submenu level: ***/system backup***

Description

The **save** command is used to store the entire router configuration in a backup file. The file is shown in the **/file** submenu. It can be downloaded via ftp to keep it as a backup for your configuration.

To restore the system configuration, for example, after a **/system reset**, it is possible to upload that file via ftp and load that backup file using **load** command in **/system backup** submenu.

Command Description

load name=[filename] - Load configuration backup from a file
save name=[filename] - Save configuration backup to a file

Example

To save the router configuration to file **test**:

```
[admin@HauteSpot] system backup> save name=test
Configuration backup saved
[admin@HauteSpot] system backup>
```

To see the files stored on the router:

```
[admin@HauteSpot] > file print
  # NAME                            TYPE         SIZE       CREATION-TIME
  0 test.backup                     backup       12567      sep/08/2004
21:07:50
[admin@HauteSpot] >
```

Example

To load the saved backup file **test**:

```
[admin@HauteSpot] system backup> load name=test
Restore and reboot? [y/N]: y
...
```

The Export Command

Command name: ***/export***

Description

The **export** command prints a script that can be used to restore configuration. The command can be invoked at any menu level, and it acts for that menu level and all menu levels below it. If the argument **from** is used, then it is possible to export only specified items. In this case **export** does not descend recursively through the command hierarchy. **export** also has the argument **file**, which allows you to save the script in a file on the router to retrieve it later via FTP.

Command Description

file=[filename] - saves the export to a file
from=[number] - specifies from which item to start to generate the export file

Example

```
[admin@HauteSpot] > ip address print
Flags: X - disabled, I - invalid, D - dynamic
 #   ADDRESS            NETWORK         BROADCAST       INTERFACE
 0   10.1.0.172/24      10.1.0.0        10.1.0.255      bridge1
 1   10.5.1.1/24        10.5.1.0        10.5.1.255      ether1
[admin@HauteSpot] >
```

To make an export file:

```
[admin@HauteSpot] ip address> export file=address
```

```
[admin@HauteSpot] ip address>
```

To make an export file from only one item:

```
[admin@HauteSpot] ip address> export file=address1 from=1
[admin@HauteSpot] ip address>
```

To see the files stored on the router:

```
[admin@HauteSpot] > file print
 # NAME                            TYPE         SIZE       CREATION-TIME
0  address.rsc                     script       315        dec/23/2003
13:21:48
1  address1.rsc                    script       201        dec/23/2003
13:22:57
[admin@HauteSpot] >
```

To export the setting on the display use the same command without the **file** argument:

```
[admin@HauteSpot] ip address> export from=0,1
# nov/13/2004 13:25:30 by RouterOS 2.9
# software id = MGJ4-MAN
#
/ ip address
add address=10.1.0.172/24 network=10.1.0.0 broadcast=10.1.0.255 \
    interface=bridge1 comment="" disabled=no
add address=10.5.1.1/24 network=10.5.1.0 broadcast=10.5.1.255 \
    interface=ether1 comment="" disabled=no
[admin@HauteSpot] ip address>
```

The Import Command

Command name: ***/import***

Description

The root level command **/import [file_name]** restores the exported information from the specified file. This is used to restore configuration or part of it after a **/system reset** event or anything that causes configuration data loss.

Note that it is impossible to import the whole router configuration using this feature. It can only be used to import a part of configuration (for example, firewall rules) in order to spare you some typing.

Command Description

file=[filename] - loads the exported configuration from a file to router

Example

To load the saved export file use the following command:

```
[admin@HauteSpot] > import address.rsc
Opening script file address.rsc
Script file loaded successfully
[admin@HauteSpot] >
```

Configuration Reset

Command name: ***/system reset***

Description

The command clears all configuration of the router and sets it to the default including the login name and password ('admin' and no password), IP addresses and other configuration is erased, interfaces will become disabled. After the **reset** command router will reboot.

Command Description

reset - erases router's configuration

Notes

If the router has been installed using netinstall and had a script specified as the initial configuration, the reset command executes this script after purging the configuration. To stop it doing so, you will have to reinstall the router.

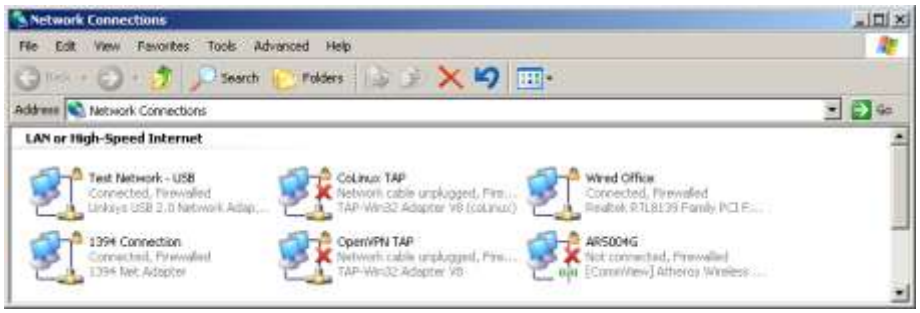
Example

```
[admin@HauteSpot] > system reset
Dangerous! Reset anyway? [y/N]: n
action cancelled
[admin@HauteSpot] >
```

Configuring Static IP on a Windows PC

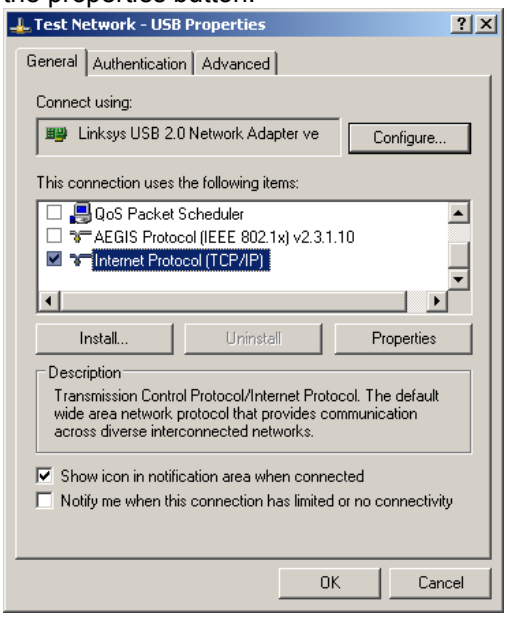
14 All HR-WRAP devices are shipped with a default IP address of 192.168.1.1 applied to the Ethernet interface. This allows a PC to connect to the device for configuration. However, by default most PC's are not configured to access this network address.

15 Go to START | SETTINGS/CONTROL PANEL | NETWORK CONNECTIONS and open that window



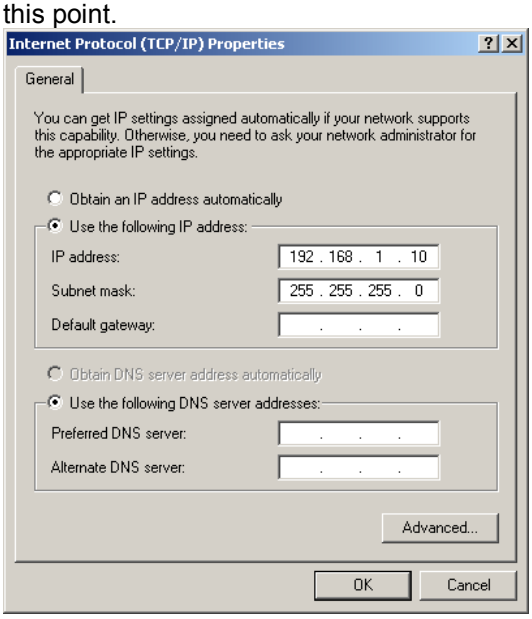
16 Right click on the network interface for your Ethernet port (typically named Local Area Network) and select properties.

17 Scroll down the window of configuration items and select Internet Protocol. Do not uncheck any of the configuration items. Highlight Internet Protocol and select the properties button.



18 Set the address to the static address of 192.168.1.10 with a subnet mask of 255.255.255.0. There is no need to set the default gateway or DNS servers at

this point.


Internet Protocol (TCP/IP) Properties
General
You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.
Obtain an IP address automatically
Use the following IP address:
IP address:
192 . 168 . 1 . 10
Subnet mask:
255 . 255 . 255 . 0
Default gateway:
Obtain DNS server address automatically
Use the following DNS server addresses:
Preferred DNS server:
Alternate DNS server:
Advanced...
OK
Cancel

19 Click OK. You are now configured to attach to a HR-WRAP piece of equipment.

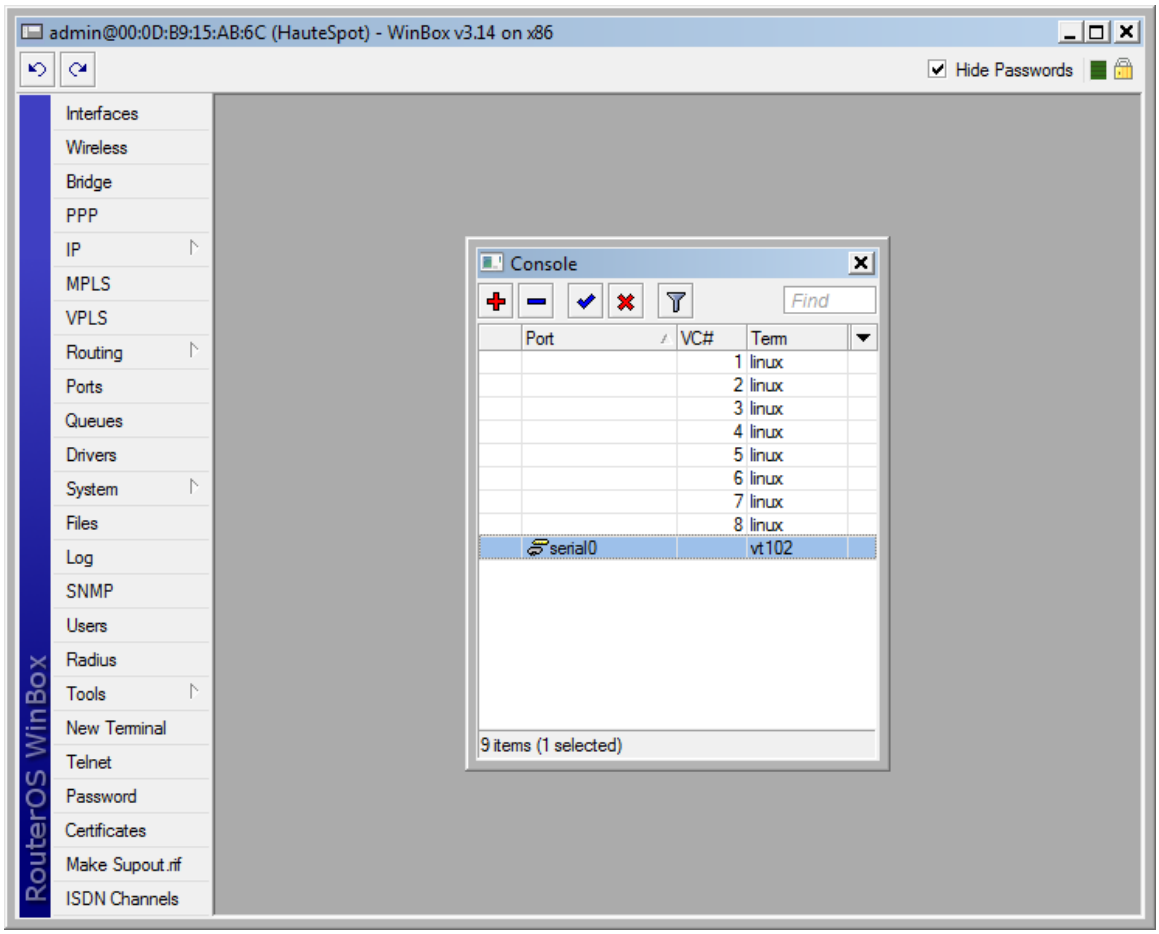
Configuring the HauteWRAP for Serial Terminal Server Operation

Serial terminal server operation allows the on-board serial port of the HauteWRAP family of products to be remotely connected to using Telnet over your network. Serial terminal server operation allows you to specify a telnet port to map to the serial port, specify the serial port configuration, and set a range of IP addresses which are permitted to connect to the serial port.

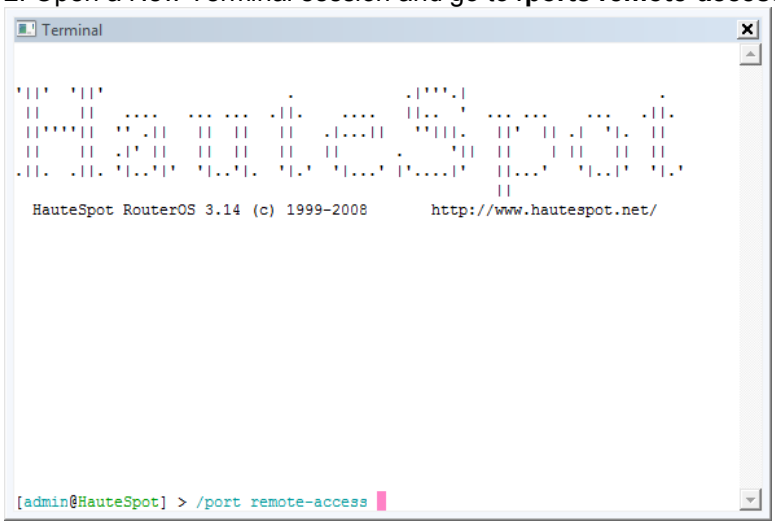
CAUTION: By using the serial terminal server function, you will be disabling local console operation on your router. This is a critical recovery feature for mis-configured systems. If you set your system to bad configuration which disables your Ethernet access to the system, you will have no way to access the system for recovery. Your only recourse will be to send your system into HauteSpot Networks for a complete factory reset. Recovery of a system that has it's serial port mis-configured is not included in your warranty and you will be charged for the labor involved in resetting your system. Use this feature with caution.

Steps:

1. Disable the use of your serial port for console access by going to the System Console menu item and deleting serial0 from the list.

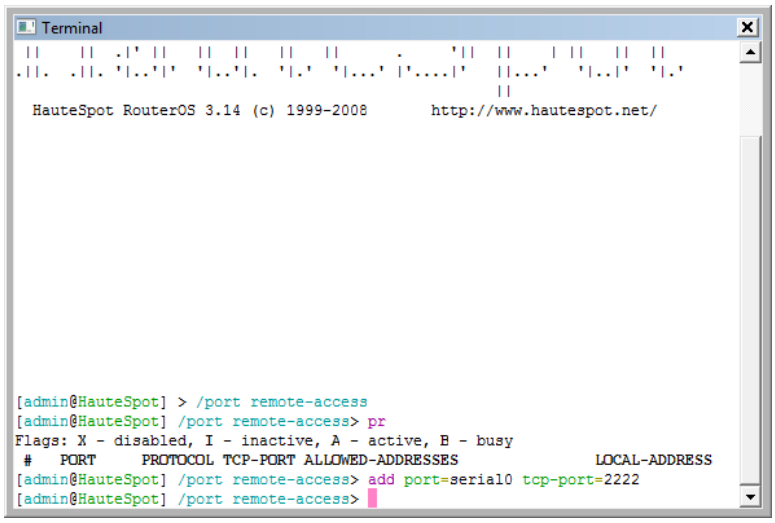


2. Open a New Terminal session and go to **/ports remote-access**.

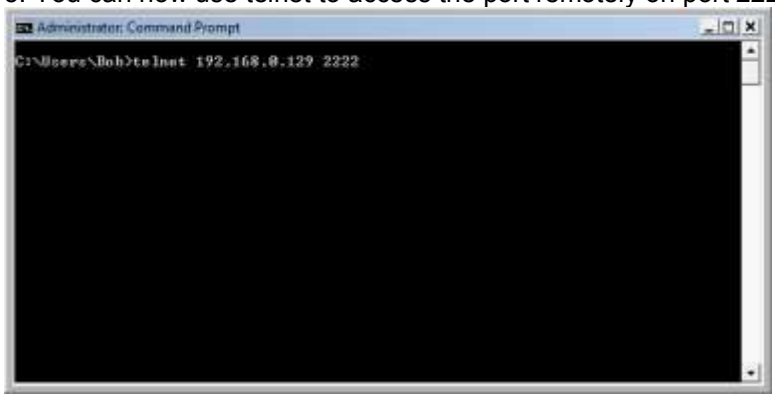


3. The **print** command will show a listing of existing Serial Terminal configurations. There should not be any.

4. Use the **add port=serial0 tcp-port=2222** command to create a new Serial Terminal Server configuration on TCP port 2222.

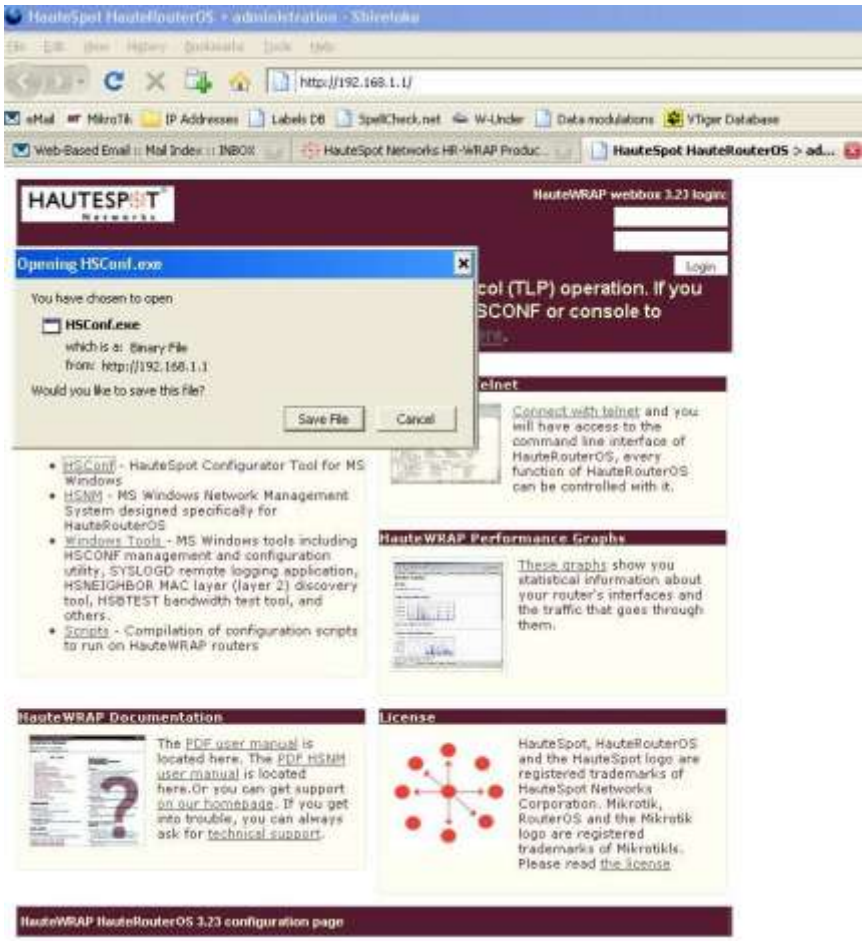

Terminal
HauteSpot RouterOS 3.14 (c) 1999-2008 http://www.hautespot.net/
[admin@HauteSpot] > /port remote-access
[admin@HauteSpot] /port remote-access> pr
Flags: X - disabled, I - inactive, A - active, B - busy
PORT PROTOCOL TCP-PORT ALLOWED-ADDRESSES LOCAL-ADDRESS
[admin@HauteSpot] /port remote-access> add port=serial0 tcp-port=2222
[admin@HauteSpot] /port remote-access>

5. You can now use telnet to access the port remotely on port 2222.



Configuring the HR-WRAPSX as a basic Bridging Access Point v4.0

20 Open your web browser on your computer to the IP address of the HR-WRAPSX (192.168.1.1 by default). Download the HSConf. tool to your desktop. If the web browser does not load the web admin you may need to assign your PC a static IP address.

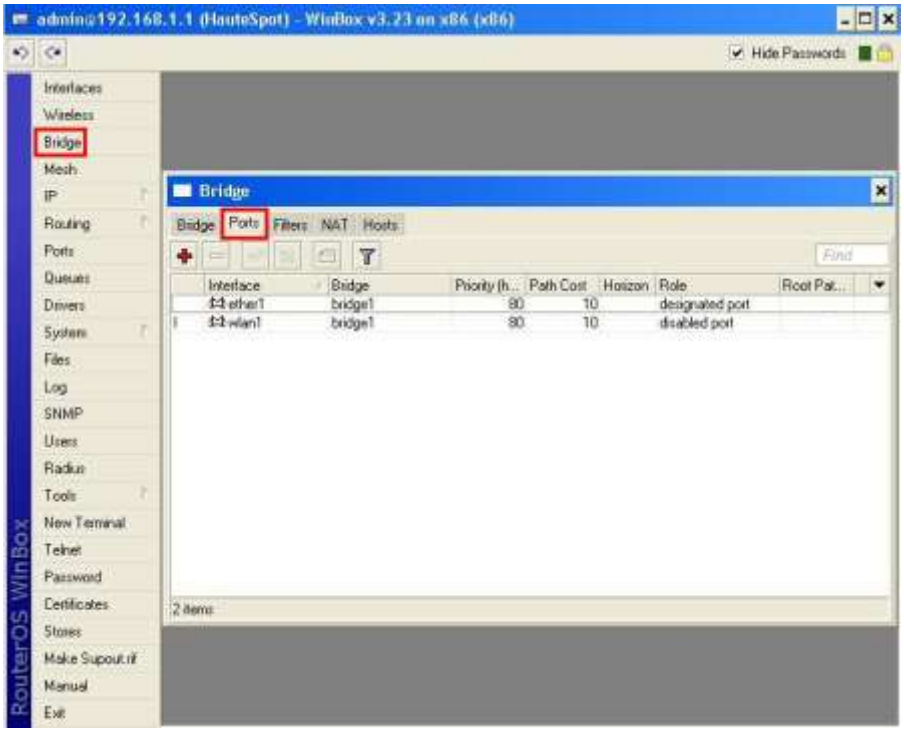

http://192.168.1.1/
HAUTESPOT
HauteWRAP webbox 3.23 login
Login
Opening HSConf.exe
You have chosen to open
HSConf.exe
which is a: Binary File
from: http://192.168.1.1
Would you like to save this file?
Save File
Cancel
col (TLP) operation. If you
SCONF or console to
HSConf - HauteSpot Configurator Tool for MS Windows
HSNM - MS Windows Network Management System designed specifically for HauteRouterOS
Windows Tools - MS Windows tools including HSCONF management and configuration utility, SYSLOGD remote logging application, HSNEIGHBOR MAC layer (layer 2) discovery tool, HSBTEST bandwidth test tool, and others.
Scripts - Compilation of configuration scripts to run on HauteWRAP routers
Connect with telnet and you will have access to the command line interface of HauteRouterOS, every function of HauteRouterOS can be controlled with it.
HauteWRAP Performance Graphs
These graphs show you statistical information about your router's interfaces and the traffic that goes through them.
HauteWRAP Documentation
The PDF user manual is located here. The PDF HSNM user manual is located here.Or you can get support on our homepage. If you get into trouble, you can always ask for technical support.
License
HauteSpot, HauteRouterOS and the HauteSpot logo are registered trademarks of HauteSpot Networks Corporation. Mikrotik, RouterOS and the Mikrotik logo are registered trademarks of Mikrotikls. Please read the license
HauteWRAP HauteRouterOS 3.23 configuration page

21 Double-click the HSConf.exe you saved to your desktop. Enter the IP address of the HR-WRAPSX. The default login is admin with no password. Select 'Connect' when fields are correct.

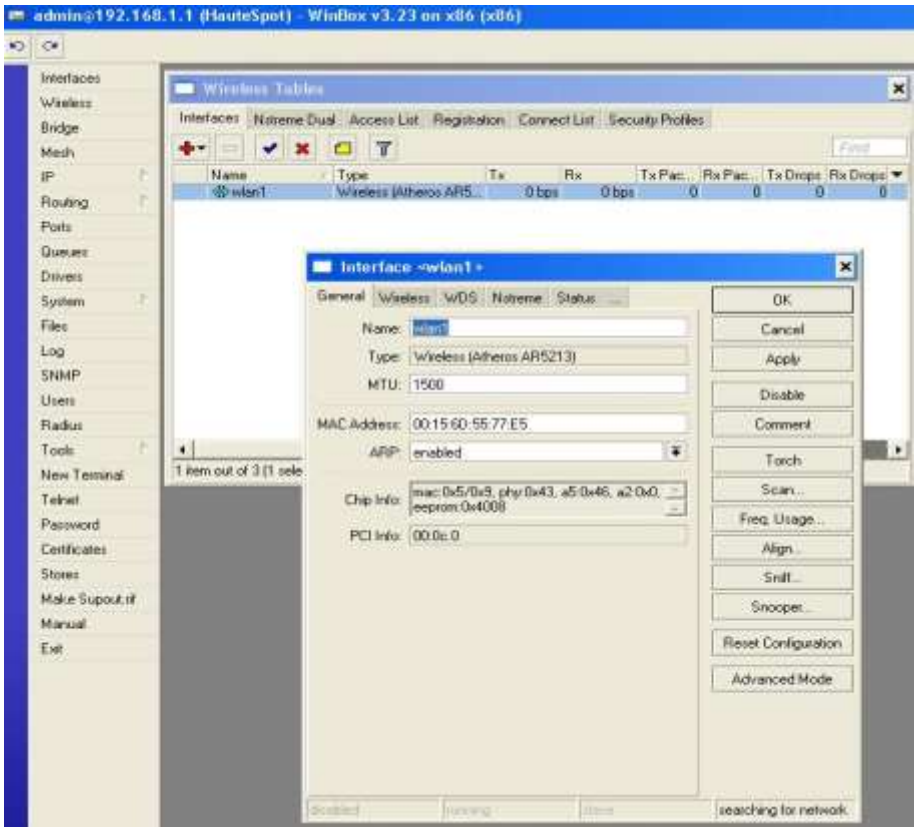
22 HSConf. may update itself. When done the main HSConf. window should be displayed:

23

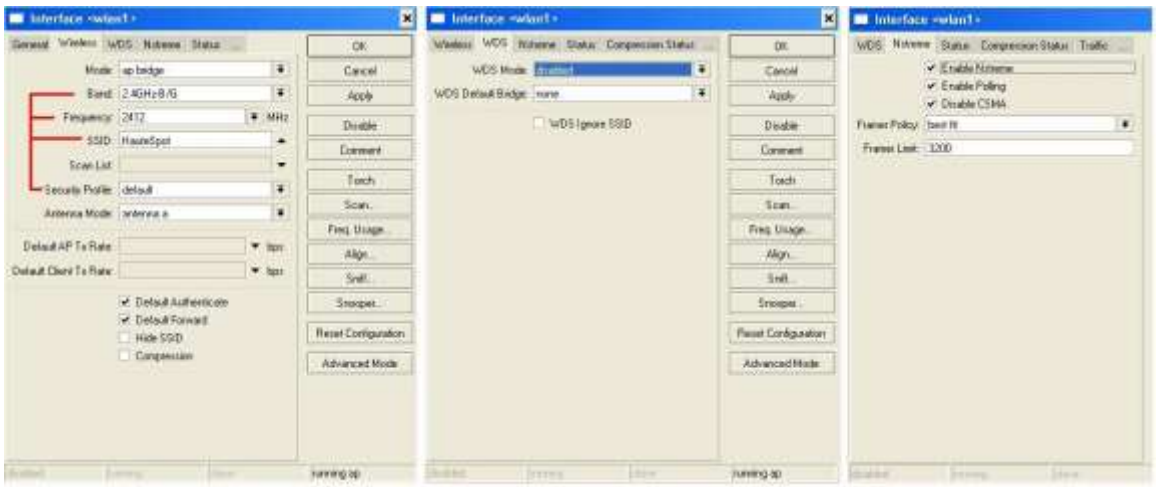
24 Now, open the bridge menu (ports tab) and verify that ether1 and wlan1 are in the bridge.



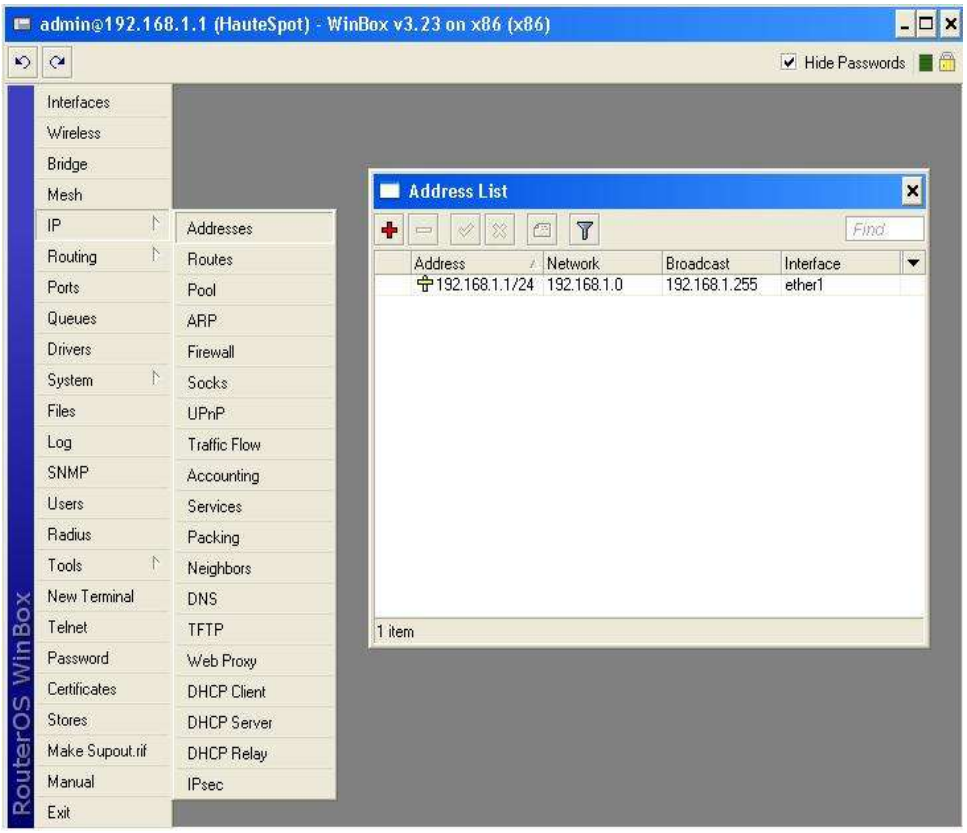
25 Close the Bridge menu and open the Wireless menu (Interfaces tab). Double click wlan1.



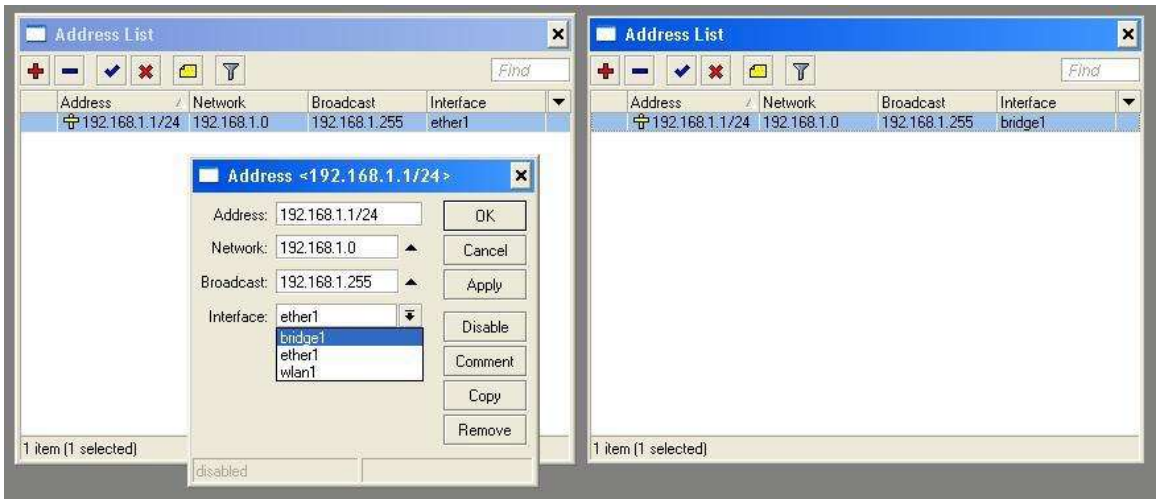
26 Most of the wireless configuration will be done in the Wirless tab of 'Interface <wlan1>' menu. However the Nstreme tab may need to be changed depending on desired performance. If operating with exclusively with HR-WRAP devices we encourage the use of our proprietary Nstreme protocol. If operating with other manufacturer equipment (latptops, pdas etc) Nstreme must be disabled. Below is a picture of a typical Wirless AP configuration. Advanced users may change the tagged. Wireless configuration should be complete at this point.



27 Lastly, we will need to change the interface the IP address is assigned to from Ether1 to Bridge1. To do this open the IP>Addresses menu.



28 Double click the existing entry and change from ether1 to bridge1.



29 Configuration is complete.

HauteRouterOS DDNS Update Tool

Document revision: 1.2 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

Dynamic DNS Update Tool gives a way to keep domain name pointing to dynamic IP address. It works by sending domain name system update request to name server, which has a zone to be updated. Secure DNS updates are also supported.

The DNS update tool supports only one algorithm - **hmac-md5**. It's the only proposed algorithm for signing DNS messages.

Specifications

Packages required: ***advanced-tools***
License required: *Level1*
Command name: ***/tool dns-update***
Standards and Technologies: Dynamic Updates in the DNS (RFC 2136), Secure DNS Dynamic Update (RFC 3007)
Hardware usage: *Not significant*

Related Documents

- Package Management

Description

Dynamic DNS Update is a tool that should be manually run to update dynamic DNS server.

Note that you have to have a DNS server that supports DNS updates properly configured.

Additional Resources

- DNS related RFCs

Dynamic DNS Update

Command name: ***/tool dns-update***

Property Description

address (*IP address*) - defines IP address associated with the domain name
dns-server (*IP address*) - DNS server to send update to
key (*text*; default: **""**) - authorization key (password of a kind) to access the server
key-name (*text*; default: **""**) - authorization key name (username of a kind) to access the server
name (*text*) - name to attach with the IP address
ttl (*integer*; default: **0**) - time to live for the item (in seconds)
zone (*text*) - DNS zone where to update the domain name in

Notes

Note that the system clock time on your router can't differ from the DNS server's time more than 5 minutes. Otherwise the DNS server will ignore this request.

Example

To tell **23.34.45.56** DNS server to (re)associate **mydomain** name in the **myzone.com** zone with **68.42.14.4** IP address specifying that the name of the key is **dns-update-key** and the actual key is **update**:

```
[admin@Hautespot] tool> dns-update dns-server=23.34.45.56 name=mydomain \
\... zone=myzone.com address=68.42.14.4 key-name=dns-update-key key=update
```

HauteRouterOS DHCP Client and Server

Document revision: 2.7 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

The DHCP (Dynamic Host Configuration Protocol) is needed for easy distribution of IP addresses in a network. The Hautespot RouterOS implementation includes both - server and client parts and is compliant with RFC2131.

General usage of DHCP:

- IP assignment in LAN, cable-modem, and wireless systems
- Obtaining IP settings on cable-modem systems

IP addresses can be bound to MAC addresses using static lease feature.

DHCP server can be used with HauteRouterOS HotSpot feature to authenticate and account DHCP clients. See the HotSpot Manual for more information.

Quick Setup Guide

This example will show you how to setup DHCP-Server and DHCP-Client on Hautespot RouterOS.

- Setup of a DHCP-Server.
    1. Create an IP address pool

```
/ip pool add name=dhcp-pool ranges=172.16.0.10-172.16.0.20
```

    2. Add a DHCP network which will concern to the network **172.16.0.0/12** and will distribute a gateway with IP address **172.16.0.1** to DHCP clients:

```
/ip dhcp-server network add address=172.16.0.0/12
gateway=172.16.0.1
```

    3. Finally, add a DHCP server:

```
/ip dhcp-server add interface=wlan1 address-pool=dhcp-pool
```

- Setup of the DHCP-Client (which will get a lease from the DHCP server, configured above).
    1. Add the DHCP client:

```
2. /ip dhcp-client add interface=wlan1 use-peer-dns=yes \
```

```
     add-default-route=yes disabled=no
```

    3. Check whether you have obtained a lease:

```
4.  [admin@Server] ip dhcp-client> print detail
5.  Flags: X - disabled, I - invalid
6.   0   interface=wlan1 add-default-route=yes use-peer-dns=yes
    status=bound
7.       address=172.16.0.20/12 gateway=172.16.0.1 dhcp-
    server=192.168.0.1
8.       primary-dns=159.148.147.194 expires-after=2d23:58:52
```

```
[admin@Server] ip dhcp-client>
```

Specifications

Packages required: ***dhcp***
License required: *Level1*
Submenu level: ***/ip dhcp-client***, ***/ip dhcp-server***, ***/ip dhcp-relay***
Standards and Technologies: DHCP

Description

The DHCP protocol gives and allocates IP addresses to IP clients. DHCP is basically insecure and should only be used in trusted networks. DHCP server always listens on UDP 67 port, DHCP client - on UDP 68 port. The initial negotiation involves communication between broadcast addresses (on some phases sender will use source address of **0.0.0.0** and/or destination address of **255.255.255.255**). You should be aware of this when building firewall.

Additional Resources

- ISC Dynamic Host Configuration Protocol (DHCP)
- DHCP mini-HOWTO
- ISC DHCP FAQ

DHCP Client Setup

Submenu level: ***/ip dhcp-client***

Description

The Hautespot RouterOS DHCP client may be enabled on any Ethernet-like interface at a time. The client will accept an address, netmask, default gateway, and two dns server addresses. The received IP address will be added to the interface with the respective netmask. The default gateway will be added to the routing table as a dynamic entry. Should the DHCP client be disabled or not renew an address, the dynamic default route will be removed. If there is already a default route installed prior the DHCP client obtains one, the route obtained by the DHCP client would be shown as invalid.

Property Description

address (*IP address/netmask*) - IP address and netmask, which is assigned to DHCP Client from the Server
add-default-route (yes | no; default: **yes**) - whether to add the default route to the gateway specified by the DHCP server
client-id (*text*) - corresponds to the settings suggested by the network administrator or ISP. Commonly it is set to the client's MAC address, but it may as well be any test string
dhcp-server (*IP address*) - IP address of the DHCP Server
enabled (yes | no; default: **no**) - whether the DHCP client is enabled
expires-after (*time*) - time, which is assigned by the DHCP Server, after which the lease expires
gateway (*IP address*) - IP address of the gateway which is assigned by DHCP Server
host-name (*text*) - the host name of the client as sent to a DHCP server
interface (*name*) - any Ethernet-like interface (this includes wireless and EoIP tunnels) on which the DHCP Client searches the DHCP Server
primary-dns (*IP address*) - IP address of the primary DNS server, assigned by the DHCP Server
secondary-dns (*IP address*) - IP address of the secondary DNS server, assigned by DHCP Server
primary-ntp - IP address of the primary NTP server, assigned by the DHCP Server
secondary-ntp - IP address of the secondary NTP server, assigned by the DHCP Server
status (bound | error | rebinding... | renewing... | requesting... | searching... | stopped) - shows the status of DHCP Client
use-peer-dns (yes | no; default: **yes**) - whether to accept the DNS settings advertized by DHCP server (they will be ovverriden in /ip dns submenu)
use-peer-ntp (yes | no; default: **yes**) - whether to accept the NTP settings advertized by DHCP server (they will override the settings put in the **/system ntp client** submenu)

Command Description

release - release current binding and restart DHCP client
renew - renew current leases. If the renew operation was not successful, client tries to reinitialize lease (i.e. it starts lease request procedure (rebind) as if it had not received an IP address yet)

Notes

If **host-name** property is not specified, client's system identity will be sent in the respective field of DHCP request.

If **client-id** property is not specified, client's MAC address will be sent in the respective field of DHCP request.

If **use-peer-dns** property is enabled, the DHCP client will unconditionally rewrite the settings in **/ip dns** submenu. In case two or more DNS servers were received, first two of them are set as primary and secondary servers respectively. In case one DNS server was received, it is put as primary server, and the secondary server is left intact.

Example

To add a DHCP client on **ether1** interface:

```
/ip dhcp-client add interface=ether1 disabled=no
[admin@Hautespot] ip dhcp-client> print detail
Flags: X - disabled, I - invalid
 0   interface=ether1 add-default-route=no use-peer-dns=no status=bound
     address=192.168.25.100/24 dhcp-server=10.10.10.1 expires-after=2d21:25:12
[admin@Hautespot] ip dhcp-client>
```

DHCP Server Setup

Submenu level: ***/ip dhcp-server***

Description

The router supports an individual server for each Ethernet-like interface. The Hautespot RouterOS DHCP server supports the basic functions of giving each requesting client an IP address/netmask lease, default gateway, domain name, DNS-server(s) and WINS-server(s) (for Windows clients) information (set up in the DHCP networks submenu)

In order DHCP server to work, you must set up also IP pools (do not include the DHCP server's IP address into the pool range) and DHCP networks.

It is also possible to hand out leases for DHCP clients using the RADIUS server, here are listed the parameters for used in RADIUS server.

Access-Request:

- **NAS-Identifier** - router identity
- **NAS-IP-Address** - IP address of the router itself
- **NAS-Port** - unique session ID
- **NAS-Port-Type** - Ethernet
- **Calling-Station-Id** - client identifier (active-client-id)
- **Framed-IP-Address** - IP address of the client (active-address)
- **Called-Station-Id** - name of DHCP server
- **User-Name** - MAC address of the client (active-mac-address)
- **Password** - ""

Access-Accept:

- **Framed-IP-Address** - IP address that will be assigned to client
- **Framed-Pool** - ip pool from which to assign ip address to client
- **Rate-Limit** - Datarate limitation for DHCP clients. Format is: rx-rate[/tx-rate] [rx-burst-rate[/tx-burst-rate] [rx-burst-threshold[/tx-burst-threshold] [rx-burst-time[/tx-burst-time][priority] [rx-rate-min[/tx-rate-min]]]]. All rates should be numbers with optional 'k' (1,000s) or 'M' (1,000,000s). If tx-rate is not specified, rx-rate is as tx-rate too. Same goes for tx-burst-rate and tx-burst-threshold and tx-burst-time. If both rx-burst-threshold and tx-burst-threshold are not specified (but burst-rate is specified), rx-rate and tx-rate are used as burst thresholds. If both rx-burst-time and tx-burst-time are not specified, 1s is used as default. Priority takes values 1..8, where 1 implies the highest priority, but 8 - the lowest. If rx-rate-min and tx-rate-min are not specified rx-rate and tx-rate values are used. The rx-rate-min and tx-rate-min values can not exceed rx-rate and tx-rate values.
- **Ascend-Data-Rate** - tx/rx data rate limitation if multiple attributes are provided, first limits tx data rate, second - rx data rate. If used together with Ascend-Xmit-Rate, specifies rx rate. 0 if unlimited
- **Ascend-Xmit-Rate** - tx data rate limitation. It may be used to specify tx limit only instead of sending two sequental Ascend-Data-Rate attributes (in that case Ascend-Data-Rate will specify the receive rate). 0 if unlimited
- **Session-Timeout** - max lease time (lease-time)

Property Description

add-arp (yes | no; default: **no**) - whether to add dynamic ARP entry:
no - either ARP mode should be **enabled** on that interface or static ARP entries should be administratively defined in **/ip arp** submenu
address-pool (name | static-only; default: **static-only**) - IP pool, from which to take IP addresses for clients
static-only - allow only the clients that have a static lease (i.e. no dynamic addresses will be given to clients, only the ones added in **lease** submenu)
always-broadcast (yes | no; default: **no**) - always send replies as broadcasts
authoritative (after-10sec-delay | after-2sec-delay | no | yes; default: **after-2sec-delay**) - whether the DHCP server is the only one DHCP server for the network
after-10sec-delay - to clients request for an address, dhcp server will wait 10 seconds and if there is another request from the client after this period of time, then dhcp server will offer the address to the client or will send DHCPNAK, if the requested address is not available from this server
after-2sec-delay - to clients request for an address, dhcp server will wait 2 seconds and if there is another request from the client after this period of time, then dhcp server will offer the address to the client or will send DHCPNAK, if the requested address is not available from this server
no - dhcp server ignores clients requests for addresses that are not available from this server
yes - to clients request for an address that is not available from this server, dhcp server will send negative acknowledgment (DHCPNAK)
bootp-support (none | static | dynamic; default: **static**) - support for BOOTP clients
none - do not respond to BOOTP requests
static - offer only static leases to BOOTP clients
dynamic - offer static and dynamic leases for BOOTP clients
delay-threshold (*time*; default: **none**) - if secs field in DHCP packet is smaller than *delay-threshold*, then this packet is ignored

none - there is no threshold (all DHCP packets are processed)
interface (*name*) - Ethernet-like interface name
lease-time (*time*; default: **72h**) - the time that a client may use an address. The client will try to renew this address after a half of this time and will request a new address after time limit expires
name (*name*) - reference name
ntp-server (*text*) - the DHCP client will use these as the default NTP servers. Two comma-separated NTP servers can be specified to be used by DHCP client as primary and secondary NTP servers
relay (*IP address*; default: **0.0.0.0**) - the IP address of the relay this DHCP server should process requests from:
0.0.0.0 - the DHCP server will be used only for direct requests from clients (no DHCP really allowed)
255.255.255.255 - the DHCP server should be used for any incomming request from a DHCP relay except for those, which are processed by another DHCP server that exists in the **/ip dhcp-server** submenu
src-address (*IP address*; default: **0.0.0.0**) - the address which the DHCP client must send requests to in order to renew an IP address lease. If there is only one static address on the DHCP server interface and the source-address is left as **0.0.0.0**, then the static address will be used. If there are multiple addresses on the interface, an address in the same subnet as the range of given addresses should be used
use-radius (yes | no; default: **no**) - whether to use RADIUS server for dynamic leases

Notes

If using both - Universal Client and DHCP Server on the same interface, client will only receive a DHCP lease in case it is directly reachable by its MAC address through that interface (some wireless bridges may change client's MAC address).

If **authoritative** property is set to **yes**, the DHCP server is sending rejects for the leases it cannot bind or renew. It also may (although not always) help to prevent the users of the network to run illicitly their own DHCP servers disturbing the proper way this network should be functioning.

If **relay** property of a DHCP server is not set to **0.0.0.0** the DHCP server will not respond to the direct requests from clients.

Example

To add a DHCP server to interface **ether1**, lending IP addresses from **dhcp-clients** IP pool for 2 hours:

```
/ip dhcp-server add name=dhcp-office disabled=no address-pool=dhcp-clients \
interface=ether1 lease-time=2h
[admin@Hautespot] ip dhcp-server> print
Flags: X - disabled, I - invalid
 #   NAME            INTERFACE RELAY          ADDRESS-POOL LEASE-TIME ADD-
ARP
```

```
 0   dhcp-office      ether1                    dhcp-clients 02:00:00
[admin@Hautespot] ip dhcp-server>
```

Store Leases on Disk

Submenu level: ***/ip dhcp-server config***

Description

Leases are always stored on disk on graceful shutdown and reboot. If on every lease change it is stored on disk, a lot of disk writes happen. There are no problems if it happens on a hard drive, but is very bad on Compact Flash (especially, if lease times are very short). To minimize writes on disk, all changes are flushed together every **store-leases-disk** seconds. If this time will be very short (immediately), then no changes will be lost even in case of hard reboots and power losts. But, on CF there may be too many writes in case of short lease times (as in case of hotspot). If this time will be very long (never), then there will be no writes on disk, but information about active leases may be lost in case of power loss. In these cases dhcp server may give out the same ip address to another client, if first one will not respond to ping requests.

Property Description

store-leases-disk (time-interval | immediately | never; default: **5min**) - how frequently lease changes should be stored on disk

DHCP Networks

Submenu level: ***/ip dhcp-server network***

Property Description

address (*IP address/netmask*) - the network DHCP server(s) will lend addresses from
boot-file-name (*text*) - Boot file name
dhcp-option (*text*) - add additional DHCP options from */ip dhcp-server option* list. You cannot redefine parameters which are already defined in this submenu:
Subnet-Mask (code 1) - *netmask*
Router (code 3) - *gateway*
Domain-Server (code 6) - *dns-server*
Domain-Name (code 15) - *domain*
NETBIOS-Name-Server - *wins-server*
dns-server (*text*) - the DHCP client will use these as the default DNS servers. Two comma-separated DNS servers can be specified to be used by DHCP client as primary and secondary DNS servers

domain (*text*) - the DHCP client will use this as the 'DNS domain' setting for the network adapter
gateway (*IP address*; default: **0.0.0.0**) - the default gateway to be used by DHCP clients
netmask (*integer*: 0..32; default: **0**) - the actual network mask to be used by DHCP client
0 - netmask from network **address** is to be used
next-server (*IP address*) - IP address of next server to use in bootstrap
wins-server (*text*) - the Windows DHCP client will use these as the default WINS servers. Two comma-separated WINS servers can be specified to be used by DHCP client as primary and secondary WINS servers

Notes

The **address** field uses netmask to specify the range of addresses the given entry is valid for. The actual netmask clients will be using is specified in **netmask** property.

DHCP Server Leases

Submenu level: ***/ip dhcp-server lease***

Description

DHCP server lease submenu is used to monitor and manage server's leases. The issued leases are showed here as dynamic entries. You can also add static leases to issue the definite client (determined by MAC address) the specified IP address.

Generally, the DHCP lease it allocated as follows:

1. an unused lease is in **waiting** state
2. if a client asks for an IP address, the server chooses one
3. if the client will receive statically assigned address, the lease becomes **offered**, and then **bound** with the respective lease time
4. if the client will receive a dynamic address (taken from an IP address pool), the router sends a ping packet and waits for answer for 0.5 seconds. During this time, the lease is marked **testing**
5. in case, the address does not respond, the lease becomes **offered**, and then **bound** with the respective lease time
6. in other case, the lease becomes **busy** for the lease time (there is a command to retest all busy addresses), and the client's request remains unanswered (the client will try again shortly)

A client may free the leased address. When the dynamic lease is removed, and the allocated address is returned to the address pool. But the static lease becomes **busy** until the client will reacquire the address.

Note that the IP addresses assigned statically are not probed.

Property Description

active-address (*read-only: IP address*) - actual IP address for this lease
active-client-id (*read-only: text*) - actual client-id of the client
active-mac-address (*read-only: MAC address*) - actual MAC address of the client
active-server (*read-only: list*) - actual dhcp server, which serves this client
address (*IP address*) - specify ip address (or ip pool) for static lease
0.0.0.0 - use pool from server
agent-circuit-id (*read-only: text*) - circuit ID of DHCP relay agent
agent-remote-id (*read-only: text*) - Remote ID, set by DHCP relay agent
block-access (yes | no; default: **no**) - block access for this client (drop packets from this client)
client-id (*text*; default: **""**) - if specified, must match DHCP 'client identifier' option of the request
expires-after (*read-only: time*) - time until lease expires
host-name (*read-only: text*) - shows host name option from last received DHCP request
lease-time (*time*; default: **0s**) - time that the client may use an address
0s - lease will never expire
mac-address (*MAC address*; default: **00:00:00:00:00:00**) - if specified, must match MAC address of the client
radius (*read-only:* yes | no) - shows, whether this dynamic lease is authenticated by RADIUS or not
rate-limit (*read-only: text*; default: **""**) - sets rate limit for active lease. Format is: rx-rate[/tx-rate] [rx-burst-rate[/tx-burst-rate] [rx-burst-threshold[/tx-burst-threshold] [rx-burst-time[/tx-burst-time]]]]. All rates should be numbers with optional 'k' (1,000s) or 'M' (1,000,000s). If tx-rate is not specified, rx-rate is as tx-rate too. Same goes for tx-burst-rate and tx-burst-threshold and tx-burst-time. If both rx-burst-threshold and tx-burst-threshold are not specified (but burst-rate is specified), rx-rate and tx-rate is used as burst thresholds. If both rx-burst-time and tx-burst-time are not specified, 1s is used as default.
rx-rate (*integer*; default: **0**) - maximal receive bitrate to the client (for users it is upload bitrate))
0 - no limitation
server (*read-only: name*) - server name which serves this client
status (*read-only:* waiting | testing | authorizing | busy | offered | bound) - lease status:
waiting - not used static lease
testing - testing whether this address is used or not (only for dynamic leases) by pinging it with timeout of 0.5s
authorizing - waiting for response from radius server
busy - this address is assigned statically to a client or already exists in the network, so it can not be leased
offered - server has offered this lease to a client, but did not receive confirmation from the client
bound - server has received client's confirmation that it accepts offered address, it is using it now and will free the address not later, than the lease time will be over
tx-rate (*integer*; default: **0**) - maximal transmit bitrate to the client (for users it is download bitrate))
0 - no limitation

Command Description

check-status - Check status of a given busy dynamic lease, and free it in case of no response
make-static - convert a dynamic lease to static one

Notes

If **rate-limit** is specified, a simple queue is added with corresponding parameters when lease enters bound state. Arp entry is added right after adding of queue is done (only if add-arp is enabled for dhcp server). To be sure, that client cannot use his ip address without getting dhcp lease and thus avoiding rate-limit, reply-only mode must be used on that ethernet interface.

Even though client address may be changed (with adding a new item) in **lease print** list, it will not change for the client. It is true for any changes in the DHCP server configuration because of the nature of the DHCP protocol. Client tries to renew assigned IP address only when half a lease time is past (it tries to renew several times). Only when full lease time is past and IP address was not renewed, new lease is asked (rebind operation).

the deault **mac-address** value will never work! You should specify a correct MAC address there.

Example

To assign 10.5.2.100 static IP address for the existing DHCP client (shown in the lease table as item #0):

```
[admin@Hautespot] ip dhcp-server lease> print
Flags: X - disabled, H - hotspot, D - dynamic
  #   ADDRESS         MAC-ADDRESS       EXPIRES-AFTER SERVER         STATUS
  0 D 10.5.2.90       00:04:EA:C6:0E:40 1h48m59s      switch         bound
  1 D 10.5.2.91       00:04:EA:99:63:C0 1h42m51s      switch         bound
[admin@Hautespot] ip dhcp-server lease> add copy-from=0 address=10.5.2.100
[admin@Hautespot] ip dhcp-server lease> print
Flags: X - disabled, H - hotspot, D - dynamic
  #   ADDRESS         MAC-ADDRESS       EXPIRES-AFTER SERVER         STATUS
  1 D 10.5.2.91       00:04:EA:99:63:C0 1h42m18s      switch         bound
  2   10.5.2.100      00:04:EA:C6:0E:40 1h48m26s      switch         bound
[admin@Hautespot] ip dhcp-server lease>
```

DHCP Alert

Submenu level: ***/ip dhcp-server alert***

Description

To find any rogue DHCP servers as soon as they appear in your network, DHCP Alert tool can be used. It will monitor ethernet for all DHCP replies and check, whether this reply comes from a valid DHCP server. If reply from unknown DHCP server is detected, alert gets triggered:

```
[admin@Hautespot] ip dhcp-server alert>/log print
00:34:23 dhcp,critical,error,warning,info,debug dhcp alert on Public:
    discovered unknown dhcp server, mac 00:02:29:60:36:E7, ip 10.5.8.236
[admin@Hautespot] ip dhcp-server alert>
```

When the system alerts about a rogue DHCP server, it can execute a custom script.

As DHCP replies can be unicast, rogue dhcp detector may not receive any offer to other dhcp clients at all. To deal with this, rogue dhcp server acts as a dhcp client as well - it sends out dhcp discover requests once a minute

Property Description

alert-timeout (none/*time*; default: **none**) - time, after which alert will be forgotten. If after that time the same server will be detected, new alert will be generated
none - infinite time
interface (*name*) - interface, on which to run rogue DHCP server finder
invalid-server (*read-only: text*) - list of MAC addresses of detected unknown DHCP servers. Server is removed from this list after *alert-timeout*
on-alert (*text*) - script to run, when an unknown DHCP server is detected
valid-server (*text*) - list of MAC addresses of valid DHCP servers

Notes

All alerts on an interface can be cleared at any time using command: **/ip dhcp-server alert reset-alert <interface>**

Note, that e-mail can be sent, using */system logging action add target=email*

DHCP Option

Submenu level: ***/ip dhcp-server option***

Description

With help of DHCP Option, it is possible to define additional custom options for DHCP Server.

Property Description

code (*integer*: 1..254) - dhcp option code. All codes are available at http://www.iana.org/assignments/bootp-dhcp-parameters
name (*name*) - descriptive name of the option
value (*text*) - parameter's value in form of a string. If the string begins with "0x", it is assumed as a hexadecimal value

Notes

The defined options you can use in */ip dhcp-server network* submenu

According to the DHCP protocol, a parameter is returned to the DHCP client only if it requests this parameter, specifying the respective code in DHCP request Parameter-List (code 55) attribute. If the code is not included in Parameter-List attribute, DHCP server will not send it to the DHCP client.

Example

This example shows how to set DHCP server to reply on DHCP client's Hostname request (code 12) with value **Host-A**.

Add an option named **Option-Hostname** with code **12** (Hostname) and value **Host-A**:

```
[admin@Hautespot] ip dhcp-server option> add name=Hostname code=12 \
value="Host-A"
[admin@Hautespot] ip dhcp-server option> print
 # NAME                                CODE VALUE
 0 Option-Hostname                     12   Host-A
[admin@Hautespot] ip dhcp-server option>
```

Use this option in DHCP server network list:

```
[admin@Hautespot] ip dhcp-server network> add address=10.1.0.0/24 \
\... gateway=10.1.0.1 dhcp-option=Option-Hostname dns-server=159.148.60.20
[admin@Hautespot] ip dhcp-server network> print detail
 0 address=10.1.0.0/24 gateway=10.1.0.1 dns-server=159.148.60.20
   dhcp-option=Option-Hostname
[admin@Hautespot] ip dhcp-server network>
```

Now the DHCP server will reply with its Hostname **Host-A** to DHCP client (if requested)

DHCP Relay

Submenu level: ***/ip dhcp-relay***

Description

DHCP Relay is just a proxy that is able to receive a DHCP request and resend it to the real DHCP server

Property Description

dhcp-server (*text*) - list of DHCP servers' IP addresses which should the DHCP requests be forwarded to
delay-threshold (*time*; default: **none**) - if secs field in DHCP packet is smaller than delay-threshold, then this packet is ignored
interface (*name*) - interface name the DHCP relay will be working on
local-address (*IP address*; default: **0.0.0.0**) - the unique IP address of this DHCP relay needed for DHCP server to distinguish relays:
0.0.0.0 - the IP address will be chosen automatically
name (*name*) - descriptive name for relay

Notes

DHCP relay does not choose the particular DHCP server in the dhcp-server list, it just sent to all the listed servers.

Example

To add a DHCP relay named **relay** on **ether1** interface resending all received requests to the **10.0.0.1** DHCP server:

```
[admin@Hautespot] ip dhcp-relay> add name=relay interface=ether1 \
\... dhcp-server=10.0.0.1 disabled=no
[admin@Hautespot] ip dhcp-relay> print
Flags: X - disabled, I - invalid
  #   NAME                            INTERFACE DHCP-SERVER     LOCAL-ADDRESS
  0   relay                           ether1    10.0.0.1        0.0.0.0

[admin@Hautespot] ip dhcp-relay>
```

Question&Answer-Based Setup

Command name: ***/ip dhcp-server setup***

Questions

addresses to give out (*text*) - the pool of IP addresses DHCP server should lease to the clients

dhcp address space (*IP address/netmask*; default: **192.168.0.0/24**) - network the DHCP server will lease to the clients
dhcp relay (*IP address*; default: **0.0.0.0**) - the IP address of the DHCP relay between the DHCP server and the DHCP clients
dhcp server interface (*name*) - interface to run DHCP server on
dns servers (*IP address*) - IP address of the appropriate DNS server to be propagated to the DHCP clients
gateway (*IP address*; default: **0.0.0.0**) - the default gateway of the leased network
lease time (*time*; default: **3d**) - the time the lease will be valid

Notes

Depending on current settings and answers to the previous questions, default values of following questions may be different. Some questions may disappear if they become redundant (for example, there is no use of asking for 'relay' when the server will lend the directly connected network)

Example

To configure DHCP server on **ether1** interface to lend addresses from 10.0.0.2 to 10.0.0.254 which belong to the **10.0.0.0/24** network with **10.0.0.1** gateway and **159.148.60.2** DNS server for the time of 3 days:

```
[admin@Hautespot] ip dhcp-server> setup
Select interface to run DHCP server on

dhcp server interface: ether1
Select network for DHCP addresses

dhcp address space: 10.0.0.0/24
Select gateway for given network

gateway for dhcp network: 10.0.0.1
Select pool of ip addresses given out by DHCP server

addresses to give out: 10.0.0.2-10.0.0.254
Select DNS servers

dns servers: 159.148.60.20
Select lease time

lease time: 3d
[admin@Hautespot] ip dhcp-server>
```

The wizard has made the following configuration based on the answers above:

```
[admin@Hautespot] ip dhcp-server> print
Flags: X - disabled, I - invalid
  #   NAME            INTERFACE RELAY          ADDRESS-POOL LEASE-TIME ADD-
ARP
```

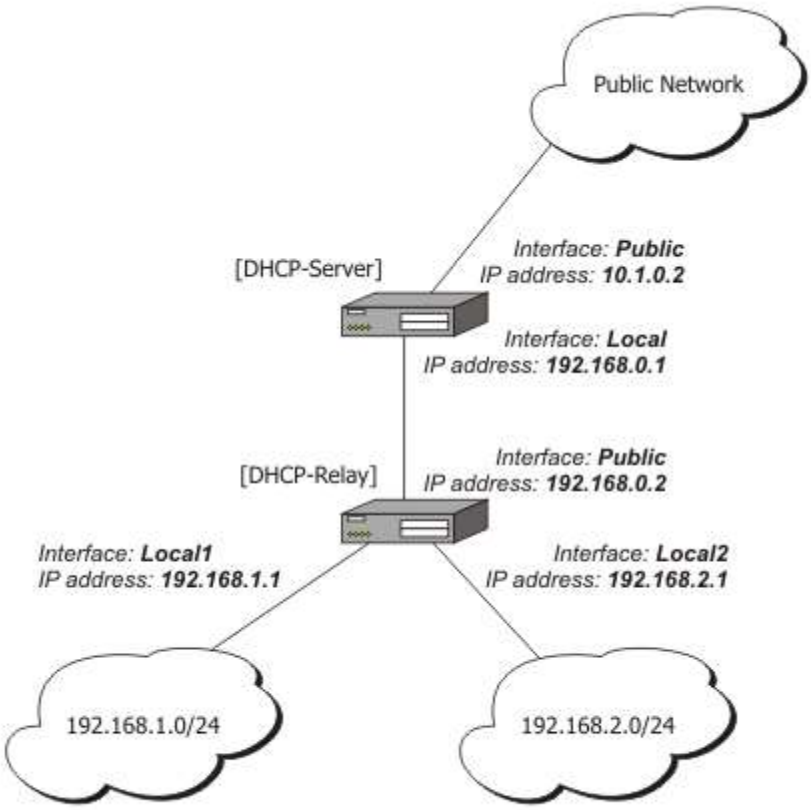
```
  0   dhcp1           ether1    0.0.0.0         dhcp_pool1    3d         no

[admin@Hautespot] ip dhcp-server> network print
  # ADDRESS            GATEWAY         DNS-SERVER      WINS-SERVER     DOMAIN
  0 10.0.0.0/24        10.0.0.1        159.148.60.20

[admin@Hautespot] ip dhcp-server> /ip pool print
  # NAME                                      RANGES
  0 dhcp_pool1                                10.0.0.2-10.0.0.254

[admin@Hautespot] ip dhcp-server>
```

Application Examples

Dynamic Addressing, using DHCP-Relay

Let us consider that you have several IP networks 'behind' other routers, but you want to keep all DHCP servers on a single router. To do this, you need a DHCP relay on your network which relies DHCP requests from clients to DHCP server.

This example will show you how to configure a DHCP server and a DHCP relay which serve 2 IP networks - **192.168.1.0/24** and **192.168.2.0/24** that are behind a router **DHCP-Relay**.



IP addresses of **DHCP-Server**:

```
[admin@DHCP-Server] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
 #   ADDRESS            NETWORK         BROADCAST       INTERFACE
 0   192.168.0.1/24     192.168.0.0     192.168.0.255   To-DHCP-Relay
 1   10.1.0.2/24        10.1.0.0 10.1.0.255       Public
[admin@DHCP-Server] ip address>
```

IP addresses of **DHCP-Relay**:

```
[admin@DHCP-Relay] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
 #   ADDRESS            NETWORK         BROADCAST       INTERFACE
 0   192.168.0.1/24     192.168.0.0     192.168.0.255   To-DHCP-Server
 1   192.168.1.1/24     192.168.1.0     192.168.1.255   Local1
 2   192.168.2.1/24     192.168.2.0     192.168.2.255   Local2
[admin@DHCP-Relay] ip address>
```

To setup 2 DHCP Servers on **DHCP-Server** router add 2 pools. For networks **192.168.1.0/24** and **192.168.2.0**:

```
/ip pool add name=Local1-Pool ranges=192.168.1.11-192.168.1.100
/ip pool add name=Local1-Pool ranges=192.168.2.11-192.168.2.100
[admin@DHCP-Server] ip pool> print
 # NAME                                      RANGES
 0 Local1-Pool                               192.168.1.11-192.168.1.100
 1 Local2-Pool                               192.168.2.11-192.168.2.100
[admin@DHCP-Server] ip pool>
```

Create DHCP Servers:

```
/ip dhcp-server add interface=To-DHCP-Relay relay=192.168.1.1 \
   address-pool=Local1-Pool name=DHCP-1 disabled=no
/ip dhcp-server add interface=To-DHCP-Relay relay=192.168.2.1 \
   address-pool=Local2-Pool name=DHCP-2 disabled=no
[admin@DHCP-Server] ip dhcp-server> print
Flags: X - disabled, I - invalid
 #   NAME         INTERFACE     RELAY           ADDRESS-POOL LEASE-TIME ADD-
ARP
 0   DHCP-1       To-DHCP-Relay 192.168.1.1     Local1-Pool  3d00:00:00
 1   DHCP-2       To-DHCP-Relay 192.168.2.1     Local2-Pool  3d00:00:00
[admin@DHCP-Server] ip dhcp-server>
```

Configure respective networks:

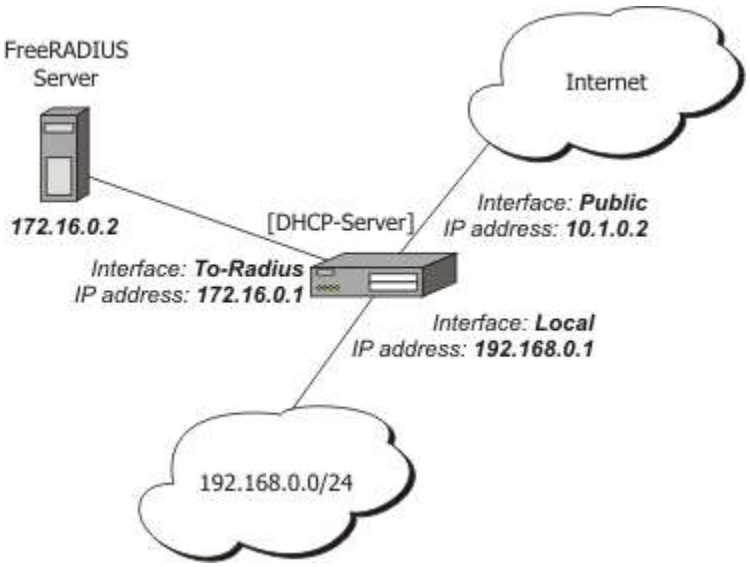
```
/ip dhcp-server network add address=192.168.1.0/24 gateway=192.168.1.1 \
   dns-server=159.148.60.20
/ip dhcp-server network add address=192.168.2.0/24 gateway=192.168.2.1 \
   dns-server 159.148.60.20
[admin@DHCP-Server] ip dhcp-server network> print
 # ADDRESS            GATEWAY         DNS-SERVER      WINS-SERVER     DOMAIN
 0 192.168.1.0/24     192.168.1.1     159.148.60.20
 1 192.168.2.0/24     192.168.2.1     159.148.60.20
[admin@DHCP-Server] ip dhcp-server network>
```

Configuration of **DHCP-Server** is done. Now let's configure **DHCP-Relay**:

```
/ip dhcp-relay add name=Local1-Relay interface=Local1 \
   dhcp-server=192.168.0.1 local-address=192.168.1.1 disabled=no
/ip dhcp-relay add name=Local2-Relay interface=Local2 \
   dhcp-server=192.168.0.1 local-address=192.168.2.1 disabled=no
[admin@DHCP-Relay] ip dhcp-relay> print
Flags: X - disabled, I - invalid
 #   NAME                        INTERFACE      DHCP-SERVER     LOCAL-ADDRESS
 0   Local1-Relay                Local1         192.168.0.1     192.168.1.1
 1   Local2-Relay                Local2         192.168.0.1     192.168.2.1
[admin@DHCP-Relay] ip dhcp-relay>
```

IP Address assignment, using FreeRADIUS Server

Let us consider that we want to assign IP addresses for clients, using the RADIUS server.



We assume that you already have installed FreeRADIUS. Just add these lines to specified files:

users file:

```
00:0B:6B:31:02:4B       Auth-Type := Local, Password == ""
        Framed-IP-Address = 192.168.0.55
```

clients.conf file

```
client 172.16.0.1 {
    secret = MySecret
    shortname = Server
}
```

Configure Radius Client on RouterOS:

```
/radius add service=dhcp address=172.16.0.2 secret=MySecret
[admin@DHCP-Server] radius> print detail
Flags: X - disabled
 0   service=dhcp called-id="" domain="" address=172.16.0.2 secret="MySecret"
     authentication-port=1812 accounting-port=1813 timeout=00:00:00.300
     accounting-backup=no realm=""
```

```
[admin@DHCP-Server] radius>
```

Setup DHCP Server:

1. Create an address pool:

```
/ip pool add name=Radius-Clients ranges=192.168.0.11-192.168.0.100
```

2. Add a DHCP server:

```
3. /ip dhcp-server add address-pool=Radius-Clients use-radius=yes
   interface=Local \
```

```
   disabled=no
```

4. Configure DHCP networks:

```
5. /ip dhcp-server network add address=192.168.0.0/24 gateway=192.168.0.1
   \
```

```
dns-server=159.148.147.194,159.148.60.20
```

Now the client with MAC address **00:0B:6B:31:02:4B** will always receive IP address **192.168.0.55**.

HauteRouterOS DNS Client and Cache

Document revision: 1.2 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

DNS cache is used to minimize DNS requests to an external DNS server as well as to minimize DNS resolution time. This is a simple recursive DNS server with local items.

Specifications

Packages required: ***system***
License required: *Level1*
Submenu level: ***/ip dns***
Standards and Technologies: DNS
Hardware usage: *Not significant*

Description

The Hautespot router with DNS cache feature enabled can be set as a primary DNS server for any DNS-compliant clients. Moreover, Hautespot router can be specified as a primary DNS server under its dhcp-server settings. When the DNS cache is enabled, the Hautespot router responds to DNS TCP and UDP requests on port 53.

Additional Resources

- http://www.freesoft.org/CIE/Course/Section2/3.htm
- http://www.networksorcery.com/enp/protocol/dns.htm
- RFC1035

Client Configuration and Cache Setup

Submenu level: ***/ip dns***

Description

DNS client is used to provide domain name resolution for router itself as well as for the P2P clients connected to the router.

Property Description

allow-remote-requests (yes | no) - specifies whether to allow network requests
cache-max-ttl (*time*; default: **1w**) - specifies maximum time-to-live for cahce records. In other words, cache records will expire after **cache-max-ttl** time.
cache-size (*integer*: 512..10240; default: **2048KiB**) - specifies the size of DNS cache in KiB
cache-used (*read-only: integer*) - displays the currently used cache size in KiB
primary-dns (*IP address*; default: **0.0.0.0**) - primary DNS server
secondary-dns (*IP address*; default: **0.0.0.0**) - secondary DNS server

Notes

If the property **use-peer-dns** under **/ip dhcp-client** is set to **yes** then **primary-dns** under **/ip dns** will change to a DNS address given by DHCP Server.

Example

To set 159.148.60.2 as the primary DNS server and allow the router to be used as a DNS server, do the following:

```
[admin@Hautespot] ip dns> set primary-dns=159.148.60.2 \
\... allow-remote-requests=yes
[admin@Hautespot] ip dns> print
            primary-dns: 159.148.60.2
          secondary-dns: 0.0.0.0
  allow-remote-requests: yes
              cache-size: 2048KiB
          cache-max-ttl: 1w
             cache-used: 17KiB
[admin@Hautespot] ip dns>
```

Cache Monitoring

Submenu level: ***/ip dns cache***

Property Description

address (*read-only: IP address*) - IP address of the host
name (*read-only: name*) - DNS name of the host
ttl (*read-only: time*) - remaining time-to-live for the record

Static DNS Entries

Submenu level: ***/ip dns static***

Description

The Hautespot RouterOS has an embedded DNS server feature in DNS cache. It allows you to link the particular domain names with the respective IP addresses and advertize these links to the DNS clients using the router as their DNS server.

Property Description

address (*IP address*) - IP address to resolve domain name with
name (*text*) - DNS name to be resolved to a given IP address
ttl (*time*) - time-to-live of the DNS record

Example

To add a static DNS entry for **www.example.com** to be resolved to **10.0.0.1** IP address:

```
[admin@Hautespot] ip dns static> add name www.example.com address=10.0.0.1
[admin@Hautespot] ip dns static> print
 # NAME                                                    ADDRESS         TTL
 0 aaa.aaa.a                                               123.123.123.123 1d
 1 www.example.com                                         10.0.0.1        1d
[admin@Hautespot] ip dns static>
```

Flushing DNS cache

Command name: ***/ip dns cache flush***

Command Description

flush - clears internal DNS cache

Example

```
[admin@Hautespot] ip dns> cache flush
[admin@Hautespot] ip dns> print
              primary-dns: 159.148.60.2
            secondary-dns: 0.0.0.0
    allow-remote-requests: yes
               cache-size: 2048 KiB
            cache-max-ttl: 1w
               cache-used: 10 KiB
[admin@Hautespot] ip dns>
```

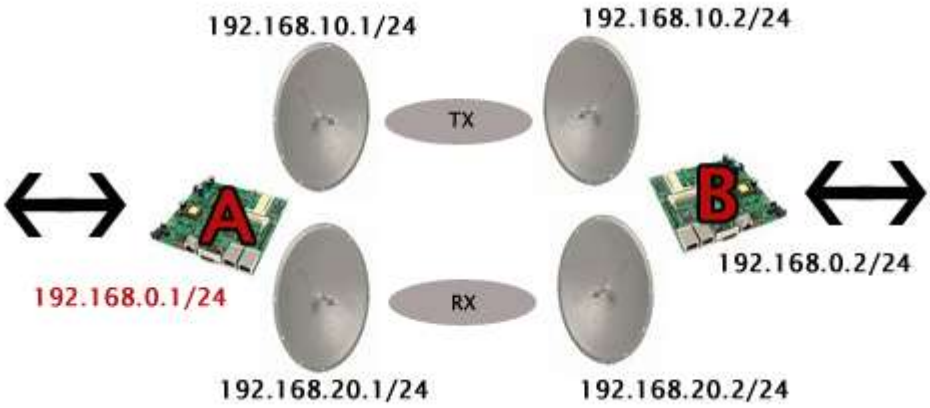
Dual Setup with OSPF

In this setup we will be connecting a PTP configuration using a Dual Setup and OSPF (Open Shortest Path First).
The requirements for this set up is to have a board that supports at least 02 radios.

The advantages for this setup are:

- Full Duplex
- Automatic Failover

- No delay of packets. The same set up utilizing Dual TDMA can cause delay, which can be a problem if you're dealing with VoIP or applications that requires maximum response.
- This set up can make use of two radios at each end, so the availability becomes even more robust in case of a radio failure.



Configuring the Radio A:

```
/ip address add address=192.168.0.1/24 interface=ether1
/ip address add address=192.168.10.1/24 interface=wlan1
/ip address add address=192.168.20.1/24 interface=wlan2
/routing ospf network add network=192.168.10.0/24 area=backbone1
/routing ospf network add network=192.168.20.0/24 area=backbone1
/routing ospf interface add interface=wlan1 cost=100
```

Configuring the Radio B:

```
/ip address add address=192.168.0.2/24 interface=ether1
/ip address add address=192.168.10.2/24 interface=wlan1
/ip address add address=192.168.20.2/24 interface=wlan2
/routing ospf network add network=192.168.10.0/24 area=backbone1
/routing ospf network add network=192.168.20.0/24 area=backbone1
/routing ospf interface add interface=wlan2 cost=100
```

This set will cause the incoming traffic to use wlan1 on *A* and wlan2 on *B*.
This also will use any available path to get to the other side (Failover with OSPF).

HauteRouterOS EoIP

Document revision: 1.4 (Tuesday, October 30, 2007)

Applies to: HauteRouterOS V3.0

General Information

Summary

Ethernet over IP (EoIP) Tunneling is a HauteRouterOS protocol that creates an Ethernet tunnel between two routers on top of an IP connection. The EoIP interface appears as an Ethernet interface. When the bridging function of the router is enabled, all Ethernet traffic (all Ethernet protocols) will be bridged just as if there where a physical Ethernet interface and cable between the two routers (with bridging enabled). This protocol makes multiple network schemes possible.

Network setups with EoIP interfaces:

- Possibility to bridge LANs over the Internet
- Possibility to bridge LANs over encrypted tunnels
- Possibility to bridge LANs over 802.11b 'ad-hoc' wireless networks

Quick Setup Guide

To make an EoIP tunnel between 2 routers which have IP addresses **10.5.8.1** and **10.1.0.1**:

1. On router with IP address **10.5.8.1**, add an EoIP interface and set its MAC address:

```
/interface eoip add remote-address=10.1.0.1 tunnel-id=1 mac-address=00-00-5E-80-00-01 \
\... disabled=no
```

3. On router with IP address **10.1.0.1**, add an EoIP interface and set its MAC address::

```
/interface eoip add remote-address=10.5.8.1 tunnel-id=1 mac-address=00-00-5E-80-00-02 \
\... disabled=no
```

Now you can add IP addresses to the created EoIP interfaces from the same subnet.

Specifications

Packages required: ***system***
License required: *Level1 (limited to 1 tunnel)* , *Level3*

Submenu level: ***/interface eoip***
Standards and Technologies: GRE (RFC1701)
Hardware usage: *Not significant*

Description

An EoIP interface should be configured on two routers that have the possibility for an IP level connection. The EoIP tunnel may run over an IPIP tunnel, a PPTP 128bit encrypted tunnel, a PPPoE connection, or any connection that transports IP.

Specific Properties:

- Each EoIP tunnel interface can connect with one remote router which has a corresponding interface configured with the same 'Tunnel ID'.
- The EoIP interface appears as an Ethernet interface under the interface list.
- This interface supports all features of an Ethernet interface. IP addresses and other tunnels may be run over the interface.
- The EoIP protocol encapsulates Ethernet frames in GRE (IP protocol number 47) packets (just like PPTP) and sends them to the remote side of the EoIP tunnel.
- Maximal count of EoIP tunnels is 65536.

Notes

WDS significantly faster than EoIP (up to 10-20% on HauteWRAP 500 systems), so it is recommended to use WDS whenever possible.

EoIP Setup

Submenu level: ***/interface eoip***

Property Description

arp (disabled | enabled | proxy-arp | reply-only; default: **enabled**) - Address Resolution Protocol
mac-address (*MAC address*) - MAC address of the EoIP interface. You can freely use MAC addresses that are in the range from **00-00-5E-80-00-00** to **00-00-5E-FF-FF-FF**
mtu (*integer*; default: **1500**) - Maximum Transmission Unit. The default value provides maximal compatibility
name (*name*; default: **eoip-tunnelN**) - interface name for reference
remote-address - the IP address of the other side of the EoIP tunnel - must be a Hautespot router
tunnel-id (*integer*) - a unique tunnel identifier

Notes

tunnel-id is method of identifying tunnel. There should not be tunnels with the same **tunnel-id** on the same router. **tunnel-id** on both participant routers must be equal.

mtu should be set to 1500 to eliminate packet refragmentation inside the tunnel (that allows transparent bridging of Ethernet-like networks, so that it would be possible to transport full-sized Ethernet frame over the tunnel).

When bridging EoIP tunnels, it is highly recommended to set unique MAC addresses for each tunnel for the bridge algorithms to work correctly. For EoIP interfaces you can use MAC addresses that are in the range from **00-00-5E-80-00-00** to **00-00-5E-FF-FF-FF**, which IANA has reserved for such cases. Alternatively, you can set the second bit of the first byte to mark the address as locally administered address, assigned by network administrator, and use any MAC address, you just need to ensure they are unique between the hosts connected to one bridge.

Example

To add and enable an EoIP tunnel named **to_mt2** to the **10.5.8.1** router, specifying **tunnel-id** of **1**:

```
[admin@Hautespot] interface eoip> add name=to_mt2 remote-address=10.5.8.1 \
\... tunnel-id 1
[admin@Hautespot] interface eoip> print
Flags: X - disabled, R - running
  0 X  name="to_mt2" mtu=1500 arp=enabled remote-address=10.5.8.1 tunnel-id=1

[admin@Hautespot] interface eoip> enable 0
[admin@Hautespot] interface eoip> print
Flags: X - disabled, R - running
  0 R  name="to_mt2" mtu=1500 arp=enabled remote-address=10.5.8.1 tunnel-id=1

[admin@Hautespot] interface eoip>
```

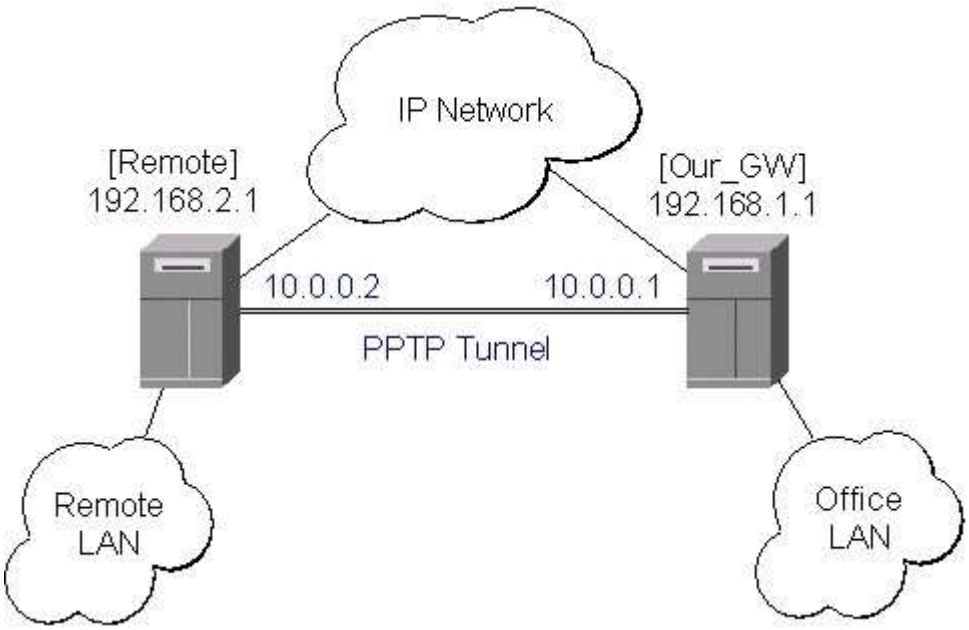
EoIP Application Example

Description

Let us assume we want to bridge two networks: 'Office LAN' and 'Remote LAN'. The networks are connected to an IP network through the routers [Our_GW] and [Remote]. The IP network can be a private intranet or the Internet. Both routers can communicate with each other through the IP network.

Example

Our goal is to create a secure channel between the routers and bridge both networks through it. The network setup diagram is as follows:



To make a secure Ethernet bridge between two routers you should:

1. Create a PPTP tunnel between them. Our_GW will be the pptp server:

```
2. [admin@Our_GW] interface pptp-server> /ppp secret add name=joe
   service=pptp \
3. \... password=top_s3 local-address=10.0.0.1 remote-address=10.0.0.2
4. [admin@Our_GW] interface pptp-server> add name=from_remote user=joe
5. [admin@Our_GW] interface pptp-server> server set enable=yes
6. [admin@Our_GW] interface pptp-server> print
7. Flags: X - disabled, D - dynamic, R - running
8.    #     NAME                 USER         MTU   CLIENT-ADDRESS  UPTIME
   ENC...
9.    0     from_remote          joe
10.[admin@Our_GW] interface pptp-server>
11.
12.The Remote router will be the pptp client:
13.
14.[admin@Remote] interface pptp-client> add name=pptp user=joe \
15.\... connect-to=192.168.1.1 password=top_s3 mtu=1500 mru=1500
16.[admin@Remote] interface pptp-client> enable pptp
17.[admin@Remote] interface pptp-client> print
18.Flags: X - disabled, R - running
19.  0  R name="pptp" mtu=1500 mru=1500 connect-to=192.168.1.1 user="joe"
20.       password="top_s2" profile=default add-default-route=no
21.
22.[admin@Remote] interface pptp-client> monitor pptp
23.      status: "connected"
24.      uptime: 39m46s
25.    encoding: "none"
26.
27.[admin@Remote] interface pptp-client>
28.
```

See the PPTP Interface Manual for more details on setting up encrypted channels.

29. Configure the EoIP tunnel by adding the eoip tunnel interfaces at both routers. Use the ip addresses of the pptp tunnel interfaces when specifying the argument values for the EoIP tunnel:

```
30. [admin@Our_GW] interface eoip> add name="eoip-remote" tunnel-id=0 \
31. \... remote-address=10.0.0.2
32. [admin@Our_GW] interface eoip> enable eoip-remote
33. [admin@Our_GW] interface eoip> print
34. Flags: X - disabled, R - running
35.   0    name=eoip-remote mtu=1500 arp=enabled remote-address=10.0.0.2
    tunnel-id=0
36. [admin@Our_GW] interface eoip>
37.
38. [admin@Remote] interface eoip> add name="eoip" tunnel-id=0 \
39. \... remote-address=10.0.0.1
40. [admin@Remote] interface eoip> enable eoip-main
41. [admin@Remote] interface eoip> print
42. Flags: X - disabled, R - running
43.   0   name=eoip mtu=1500 arp=enabled remote-address=10.0.0.1 tunnel-
    id=0
44.
```

```
[Remote] interface eoip>
```

45. Enable bridging between the EoIP and Ethernet interfaces on both routers.

    On the Our_GW:

```
[admin@Our_GW] interface bridge> add
[admin@Our_GW] interface bridge> print
Flags: X - disabled, R - running
  0  R name="bridge1" mtu=1500 arp=enabled mac-
address=00:00:00:00:00:00 stp=no
      priority=32768 ageing-time=5m forward-delay=15s
      garbage-collection-interval=4s hello-time=2s max-message-age=20s

[admin@Our_GW] interface bridge> add bridge=bridge1 interface=eoip-
remote
[admin@Our_GW] interface bridge> add bridge=bridge1 interface=office-
eth
[admin@Our_GW] interface bridge> port print
Flags: X - disabled, I - inactive, D - dynamic
 #    INTERFACE      BRIDGE  PRIORITY PATH-COST
 0    eoip-remote    bridge1 128      10
 1    office-eth     bridge1 128      10
[admin@Our_GW] interface bridge>
```

    And the same for the Remote:

```
[admin@Remote] interface bridge> add
```

```
[admin@Remote] interface bridge> print
Flags: X - disabled, R - running
  0  R name="bridge1" mtu=1500 arp=enabled mac-
address=00:00:00:00:00:00 stp=no
      priority=32768 ageing-time=5m forward-delay=15s
      garbage-collection-interval=4s hello-time=2s max-message-age=20s

[admin@Remote] interface bridge> add bridge=bridge1 interface=ether
[admin@Remote] interface bridge> add bridge=bridge1 interface=eoip-main
[admin@Remote] interface bridge> port print
Flags: X - disabled, I - inactive, D - dynamic
 #    INTERFACE      BRIDGE  PRIORITY PATH-COST
 0    ether          bridge1 128      10
 1    eoip-main      bridge1 128      10
[admin@Remote] interface bridge> port print
```

46. Addresses from the same network can be used both in the Office LAN and in the Remote LAN.

Troubleshooting

Description

- **The routers can ping each other but EoIP tunnel does not seem to work!**

  Check the MAC addresses of the EoIP interfaces - they should not be the same!

HauteRouterOS Ethernet Interfaces

Document revision: 1.2 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Specifications

Packages required: ***system***
License required: *Level1*
Submenu level: ***/interface ethernet***

Standards and Technologies: IEEE 802.3
Hardware usage: *Not significant*

Additional Resources

- http://www.ethermanage.com/ethernet/ethernet.html
- http://www.dcs.gla.ac.uk/~liddellj/nct/ethernet_protocol.html

Ethernet Interface Configuration

Submenu level: ***/interface ethernet***

Property Description

name (*name*; default: **etherN**) - assigned interface name, whrere 'N' is the number of the ethernet interface
arp (disabled | enabled | proxy-arp | reply-only; default: **enabled**) - Address Resolution Protocol
cable-setting (default | short | standard; default: **default**) - changes the cable length setting (only applicable to NS DP83815/6 cards)
default - suport long cables
short - support short cables
standard - same as default
mtu (*integer*; default: **1500**) - Maximum Transmission Unit
disable-running-check (yes | no; default: **yes**) - disable running check. If this value is set to 'no', the router automatically detects whether the NIC is connected with a device in the network or not
mac-address (*MAC address*) - set the Media Access Control number of the card
auto-negotiation (yes | no; default: **yes**) - when enabled, the interface "advertises" its maximum capabilities to achieve the best connection possible
full-duplex (yes | no; default: **yes**) - defines whether the transmission of data appears in two directions simultaneously
speed (10 Mbps | 100 Mbps | 1 Gbps) - sets the data transmission speed of the interface. By default, this value is the maximal data rate supported by the interface

Notes

For some Ethernet NICs it is possible to blink the LEDs for 10s. Type **/interface ethernet blink ether1** and watch the NICs to see the one which has blinking LEDs.

When **disable-running-check** is set to **no**, the router automatically detects whether the NIC is connected to a device in the network or not. When the remote device is not connected (the leds are not blinking), the route which is set on the specific interface, becomes invalid.

Example

```
[admin@Hautespot] > interface print
Flags: X - disabled, D - dynamic, R - running
 #    NAME                        TYPE             RX-RATE    TX-RATE    MTU
 0  X ether1                      ether            0          0          1500
[admin@Hautespot] > interface enable ether1
[admin@Hautespot] > interface print
Flags: X - disabled, D - dynamic, R - running
 #    NAME                        TYPE             RX-RATE    TX-RATE    MTU
 0  R ether1                      ether            0          0          1500
[admin@Hautespot] > interface ethernet
[admin@Hautespot] interface ethernet> print
Flags: X - disabled, R - running
 #    NAME                                  MTU   MAC-ADDRESS       ARP
 0  R ether1                                1500  00:0C:42:03:00:F2 enabled
[admin@Hautespot] interface ethernet> print detail
Flags: X - disabled, R - running
 0  R name="ether1" mtu=1500 mac-address=00:0C:42:03:00:F2 arp=enabled
      disable-running-check=yes auto-negotiation=yes full-duplex=yes
      cable-settings=default speed=100Mbps
[admin@Hautespot] interface ethernet>
```

Monitoring the Interface Status

Command name: ***/interface ethernet monitor***

Property Description

status (link-ok | no-link | unknown) - status of the interface, one of the:
link-ok - the card has connected to the network
no-link - the card has not connected to the network
unknown - the connection is not recognized
rate (10 Mbps | 100 Mbps | 1 Gbps) - the actual data rate of the connection
auto-negotiation (done | incomplete) - fast link pulses (FLP) to the adjacent link station to negotiate the SPEED and MODE of the link
done - negotiation done
incomplete - negotiation failed
full-duplex (yes | no) - whether transmission of data occurs in two directions simultaneously

Notes

See the IP Addresses and ARP section of the manual for information how to add **IP addresses** to the interfaces.

Example

```
[admin@Hautespot] interface ethernet> monitor ether1,ether2
               status: link-ok link-ok
    auto-negotiation: done    done
                rate: 100Mbps 100Mbps
```

```
        full-duplex: yes     yes
```

Troubleshooting

Description

- **Interface monitor shows wrong information**

  In some very rare cases it is possible that the device driver does not show correct information, but it does not affect the NIC's performance (of course, if your card is not broken)

Option Globetrotter HSDPA USB Modem

Introduction

This example shows how to get the Option Globetrotter HSDPA USB Modem working with two different example Wireless Service Providers. Service in your country might have different requirements and strings, but this is just to give you an outline of what is required.

Hardware

USB Modem: Option N.V. Globetrotter HSDPA USB Modem H7.2

Modem manufacturer: Teltonika, model number U3G150

Router Hardware: HauteWRAP HR-WRAPLX

Router Software

HauteRouterOS V4.2. Works in previous versions 2.9 and 3, but slightly different configuration.

The USB Modem is recognized in HauteRouterOS as an USB device and listed under the USB resources:

```
[admin@HauteSpot] > /system resource usb print
 # DEVICE VENDOR                   NAME                    SPEED
 0 2:1                             RB400 EHCI              480 Mbps
 1 1:1                             RB400 OHCI              12 Mbps
 2 1:3    Option N.V.              Globetrotter HSDPA...   12 Mbps
[admin@HauteSpot] >
```

Make sure the USB port is listed under the port menu:

```
[admin@HauteSpot] > /port print
Flags: I - inactive
 #   NAME                      CHANNELS   USED-BY
BAUD-RATE
 0   serial0                   1          Serial Console
auto
 1   usb2                      3
9600
[admin@HauteSpot] >
```

Prior HauteRouterOS V3.23, a modem had all its ports listed. Since V3.23 there is one port per modem, and modem has channels used for commands and data. Channels have numbers 0,1,2, etc. Some modems may have just two channels, some have more. Set the baud rate to 9600 for communicating with the modem, in case it's not done already:

```
[admin@HauteSpot] > /port set usb2 baud-rate=9600
```

Check the modem responses using serial-terminal, like this:

```
[admin@HauteSpot] > /system serial-terminal usb2 channel=2

[Ctrl-A is the prefix key]


ATI
Manufacturer: Option N.V.
Model: GTM378
Revision: 2.3.3Hd (Date: Jul 17 2007, Time: 15:49:23)

OK
```

Disconnect from the modem, by pressing "Ctrl-A", and then "Q":

```
[Q - quit connection]       [B - send break]
[A - send Ctrl-A prefix]    [R - autoconfigure rate]


Welcome back!
[admin@HauteSpot] >
```

Router Configuration for PPP

Since HauteRouterOS V4 (and latest versions 3) the PPP client configuration is made much easier compared to previous versions. There is no need to use ppp profiles, and ppp-client interface might have been added automatically to the /interface ppp-client list:

```
[admin@HauteSpot] > /interface ppp-client
[admin@HauteSpot] /interface ppp-client> print
Flags: X - disabled, R - running
 0 X  name="ppp-out1" max-mtu=1500 max-mru=1500 mrru=disabled port=usb2
      data-channel=2 info-channel=2 apn="internet" pin="" user=""
password=""
      profile=default phone="" dial-command="ATDT" modem-init=""
      null-modem=no dial-on-demand=yes add-default-route=yes use-peer-
dns=yes
      allow=pap,chap,mschap1,mschap2
[admin@HauteSpot] /interface ppp-client> info 0
```

The interface has 3G/GPRS modem specific arguments, like apn, pin. Consult your network provider what to use for APN, user, and password. Adjust your peer-dns and default-route settings according to what you want to do over that interface! In our case, we want permanent connection and won't use dial on demand:

```
[admin@HauteSpot] /interface ppp-client> set 0 dial-on-demand=no
[admin@HauteSpot] /interface ppp-client> enable 0
[admin@HauteSpot] /interface ppp-client> print
Flags: X - disabled, R - running
 0  R name="ppp-out1" max-mtu=1500 max-mru=1500 mrru=disabled port=usb2
      data-channel=2 info-channel=2 apn="internet" pin="" user=""
password=""
      profile=default phone="" dial-command="ATDT" modem-init=""
      null-modem=no dial-on-demand=no add-default-route=yes use-peer-
dns=yes
      allow=pap,chap,mschap1,mschap2
[admin@HauteSpot] /interface ppp-client>
```

After enabling the interface watch the logs to see how the connection is being established over the USB modem:

```
[admin@HauteSpot] /interface ppp-client> /log print
12:29:16 async,ppp,info ppp-out1: initializing...
12:29:16 async,ppp,info ppp-out1: reseting link...
12:29:16 system,info device changed by admin
12:29:16 system,info dns changed
12:29:17 async,ppp,info ppp-out1: initializing modem...
12:29:17 async,ppp,info ppp-out1: dialing out...
12:29:17 async,ppp,info ppp-out1: authenticated
12:29:20 async,ppp,info ppp-out1: could not determine remote address,
using 10.112.112.119
12:29:20 async,ppp,info ppp-out1: connected
12:29:20 system,info dns changed

[admin@HauteSpot] /interface ppp-client>
```

The serial port is used by ppp client:

```
[admin@HauteSpot] > /port print
Flags: I - inactive
 #   NAME                      CHANNELS   USED-BY
BAUD-RATE
 0   serial0                   1          Serial Console
auto
 1   usb2                      3          PPP <ppp-out1>
9600
[admin@HauteSpot] >
```

Check the addresses, in our case we have:

```
[admin@HauteSpot] > /ip address print
```

```
Flags: X - disabled, I - invalid, D - dynamic
 #   ADDRESS            NETWORK         BROADCAST       INTERFACE
 0 D 10.5.8.64/24       10.5.8.0        10.5.8.255      ether1
 1 D 10.40.192.214/32   10.112.112.119  0.0.0.0         ppp-out1
[admin@HauteSpot] >
```

and routes:

```
[admin@HauteSpot] > /ip route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        GATEWAY            DISTANCE
 0 ADS  0.0.0.0/0                          10.112.112.119     1
 1 ADC  10.5.8.0/24        10.5.8.64       ether1             0
 2 ADC  10.112.112.119/32  10.40.192.214   ppp-out1           0
[admin@HauteSpot] >
```

You can get more status information about the modem and network by running the "info" command. In our case the modem uses the same channel for data and info, thus, we disable the ppp-client and run the "info" command then:

```
[admin@HauteSpot] /interface ppp-client> info 0
failure: can't reuse channel while ppp-client running!
[admin@HauteSpot] /interface ppp-client> disable 0
[admin@HauteSpot] /interface ppp-client> info 0
             status: "ready"
         pin-status: "no password required"
      functionality: "minimum"
         gprs-class: "A - GPRS & GSM simultaneous"
       manufacturer: "Option N.V."
              model: "GTM378"
           revision: "2.3.3Hd (Date: Jul 17 2007, Time: 15:49:23)"
      serial-number: "356237010662581,S23977B51N"
   current-operator: "LV LMT GSM"
  access-technology: "3G"
     signal-strengh: -89

[admin@HauteSpot] /interface ppp-client>
```

You may need to use masquerade to hide your private network when going out through the ppp!

Modem Preferred Operation Settings

OPTION modems can be set for preferred operation in UMTS(3G) or GPRS networks. To change the operation mode, use AT_OPSYS command via terminal, or, include it into the modem init string:

```
AT_OPSYS=0,2 Only GPRS
```

```
AT_OPSYS=1,2 Only UMTS(3g)
AT_OPSYS=2,2 Prefer GPRS over UMTS(3G)
AT_OPSYS=3,2 Prefer UMTS(3G) over GPRS
```

Query the current _OPSYS setting using the AT_OPSYS? command:

```
[admin@HauteSpot] > sys serial-terminal usb4 channel=0

[Ctrl-A is the prefix key]


AT_OPSYS?
_OPSYS: 3,2

OK
```

The _OPSYS operation mode setting stays intact after router reboot or power off/on. If you lock the modem operation setting to 3G only, the scan command won't show GPRS networks, and vice versa.

Troubleshooting

1. If you do not get "authenticated" and "connected", then there is something wrong with your settings. Most likely, the ppp-client interface settings should be changed to match your provider's requirements.

2. If the data-channel setting is wrong, there would be no connection. The number of available channels is shown under the /port list, port numbers start with 0. Disable the interface and try another port number!

3. Check if you can communicate with the modem by using the serial-terminal! Make sure the ppp-client is disabled!

4. Turn off PIN request for your SIM card, it makes the life much easier. Do it in a phone if you do not know the correct AT command.

5. When speed of the link is not what expected, it's possible that the modem has connected to GPRS network, not 3G. Use the "scan" command for list of available networks:

```
[admin@HauteSpot] /interface ppp-client> scan 0
Flags: C - current, A - available, F - forbidden
  OPERATOR                    CODE       ACCESS-TECHNOLOGY
SIGNAL-STRENGH
C LV LMT GSM                  24701      GSM compact
-65
A LV LMT GSM                  24701      3G
```

```
F BITE LV                       24705       3G
F LV TELE2                      24702       GSM compact
F LV TELE2                      24702       3G
F BITE LV                       24705       GSM compact
-- [Q quit|D dump|C-z pause]
```

6. If you need to add multiple inits (multiline inits) then it is possible by dividing them using ; character. Example:

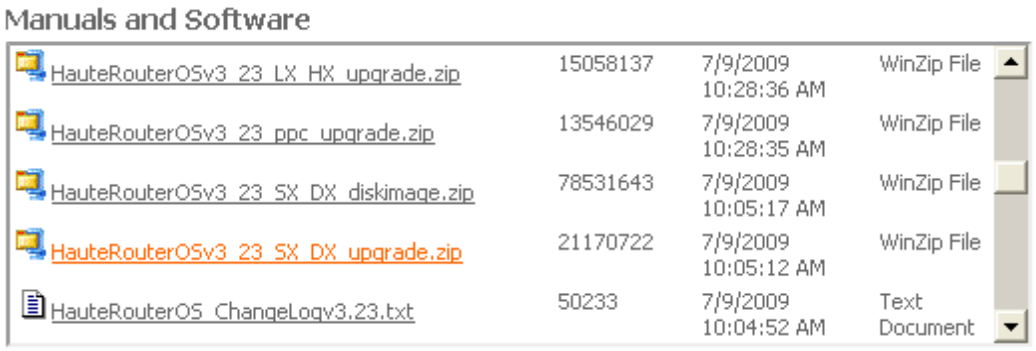
```
modem-
init="AT+CGDCONT=5,\"IP\",\"internet.lmt.l\";AT+CGDCONT=6,\"IP\",\"inte
rnet.lmt.lv\";AT"
```

Firmware update using HSconf

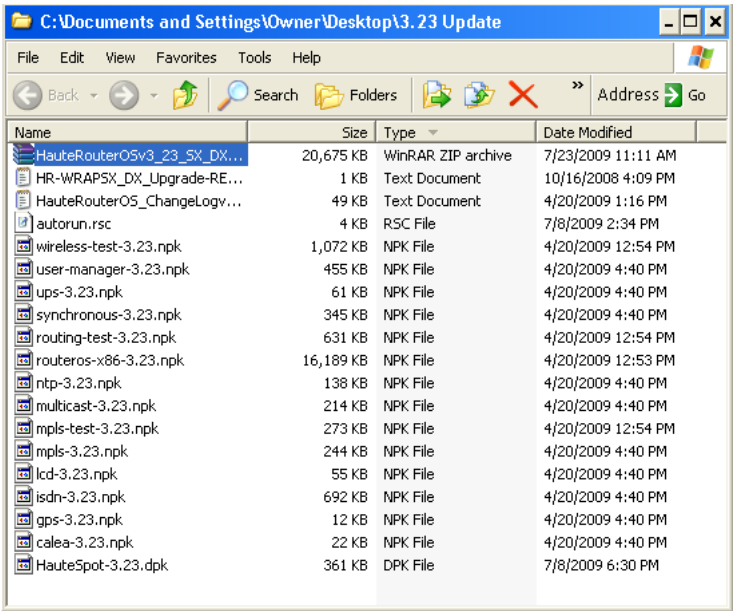
The HR-WRAP devices can have their firmware updated to introduce new features as well as repair bugs. HauteSpot Networks highly recommends that you keep your current firmware unless you are directly affected by one of these bugs or feature enhancements. *If your network is running stable avoid changes!*

HauteSpot stores all of the current firmware revisions as well as changelogs on our support page (http://www.hautespot.net/products/hr-wrap/default.asp) for the HR-WRAP device. Login requires that you register an account.

30 Begin by downloading the most current version of software from the support page. Typically there are two versions of firmware, an Update version and a Diskimage version. We will be using the Update version.

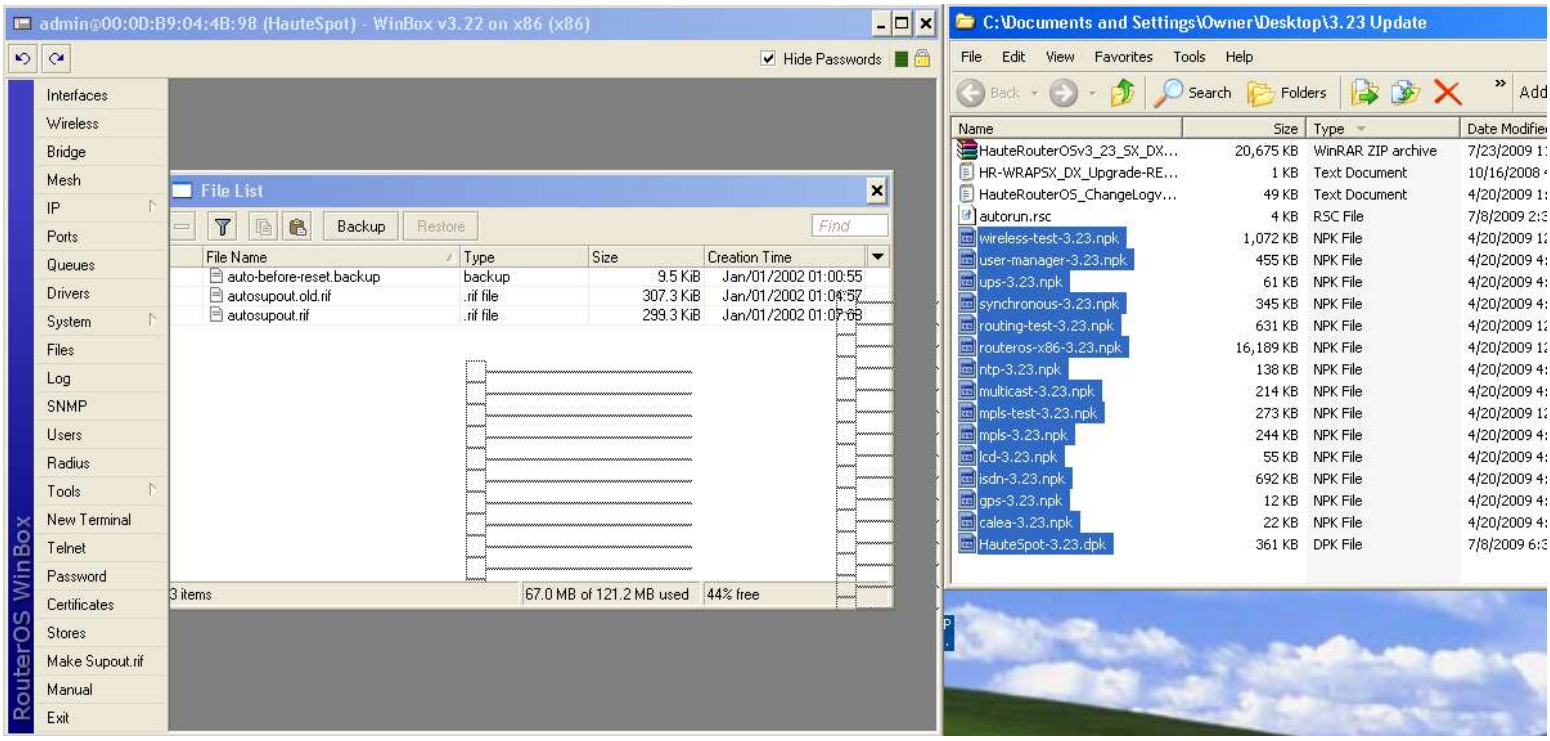

Manuals and Software
HauteRouterOSv3_23_LX_HX_upgrade.zip 15058137 7/9/2009 10:28:36 AM WinZip File
HauteRouterOSv3_23_ppc_upgrade.zip 13546029 7/9/2009 10:28:35 AM WinZip File
HauteRouterOSv3_23_SX_DX_diskimage.zip 78531643 7/9/2009 10:05:17 AM WinZip File
HauteRouterOSv3_23_SX_DX_upgrade.zip 21170722 7/9/2009 10:05:12 AM WinZip File
HauteRouterOS_ChangeLogv3.23.txt 50233 7/9/2009 10:04:52 AM Text Document

31 The downloaded package is zipped. The package will have to be unzipped into a known directory where it can be easily accessed later. The unzipped packages can be seen below.

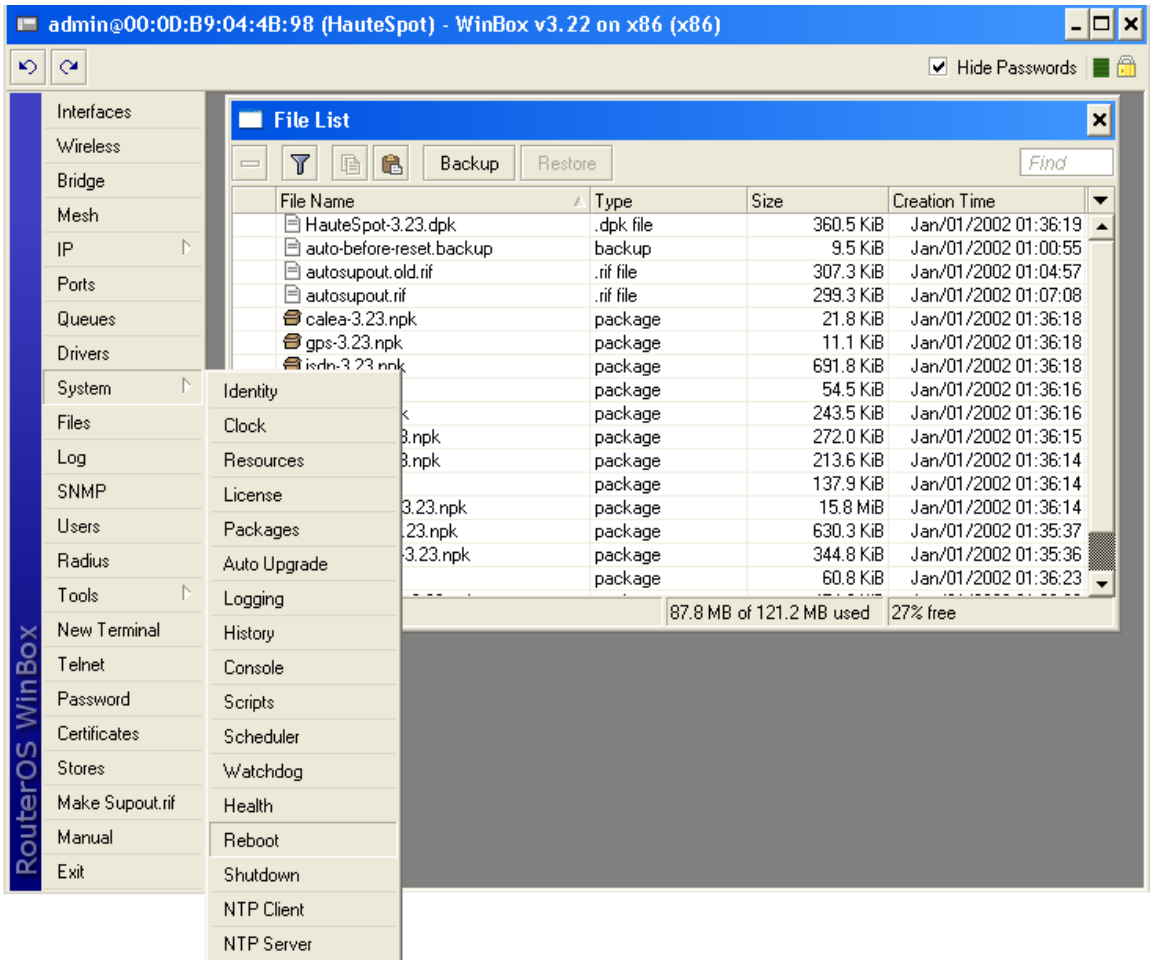

C:\Documents and Settings\Owner\Desktop\3.23 Update
File Edit View Favorites Tools Help
Back Search Folders Address Go
Name Size Type Date Modified
HauteRouterOSv3_23_SX_DX... 20,675 KB WinRAR ZIP archive 7/23/2009 11:11 AM
HR-WRAPSX_DX_Upgrade-RE... 1 KB Text Document 10/16/2008 4:09 PM
HauteRouterOS_ChangeLogv... 49 KB Text Document 4/20/2009 1:16 PM
autorun.rsc 4 KB RSC File 7/8/2009 2:34 PM
wireless-test-3.23.npk 1,072 KB NPK File 4/20/2009 12:54 PM
user-manager-3.23.npk 455 KB NPK File 4/20/2009 4:40 PM
ups-3.23.npk 61 KB NPK File 4/20/2009 4:40 PM
synchronous-3.23.npk 345 KB NPK File 4/20/2009 4:40 PM
routing-test-3.23.npk 631 KB NPK File 4/20/2009 12:54 PM
routeros-x86-3.23.npk 16,189 KB NPK File 4/20/2009 12:53 PM
ntp-3.23.npk 138 KB NPK File 4/20/2009 4:40 PM
multicast-3.23.npk 214 KB NPK File 4/20/2009 4:40 PM
mpls-test-3.23.npk 273 KB NPK File 4/20/2009 12:54 PM
mpls-3.23.npk 244 KB NPK File 4/20/2009 4:40 PM
lcd-3.23.npk 55 KB NPK File 4/20/2009 4:40 PM
isdn-3.23.npk 692 KB NPK File 4/20/2009 4:40 PM
gps-3.23.npk 12 KB NPK File 4/20/2009 4:40 PM
calea-3.23.npk 22 KB NPK File 4/20/2009 4:40 PM
HauteSpot-3.23.dpk 361 KB DPK File 7/8/2009 6:30 PM

32 Next login to the HR-WRAP device you are going to update using HSconfigurator. Note that you MUST login using the IP address of the device. Login using Mac address may result in an error during the firmware loading process. Once logged in select the 'Files' button to open the 'Files List' menu.

33 You will now need to highlight and drag all of the .npk files from your unzipped directory into the 'Files List' menu of your HR-WRAP. This will start the copying process which may take 2-3 minutes during which you should not interfere with device.

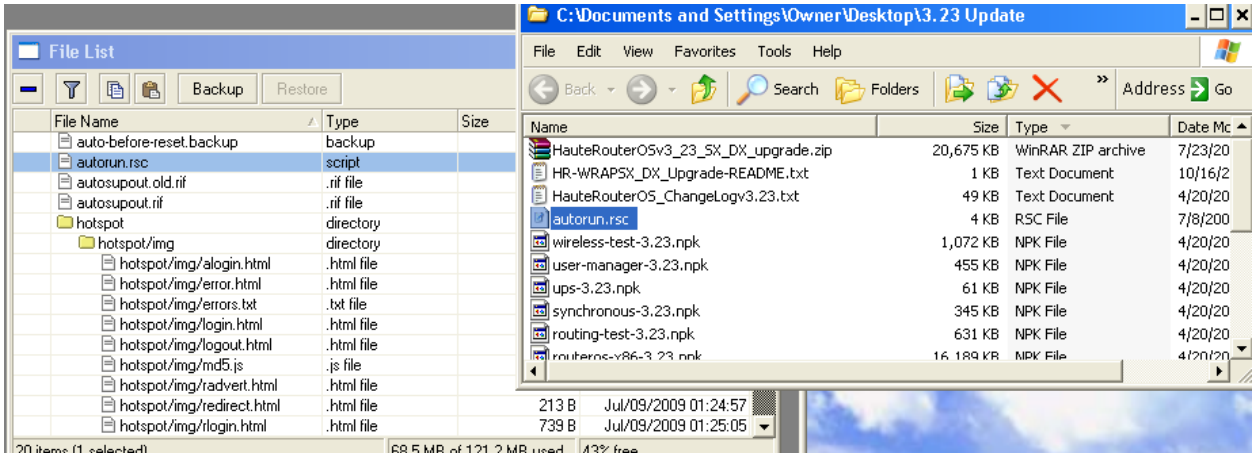


34 Once the files have completed copying you will need to restart the device. This can be done by using the 'System'->'Reboot' command in HSconfigurator. During this time power should not be removed from the device as this is when the update process begins. The update process



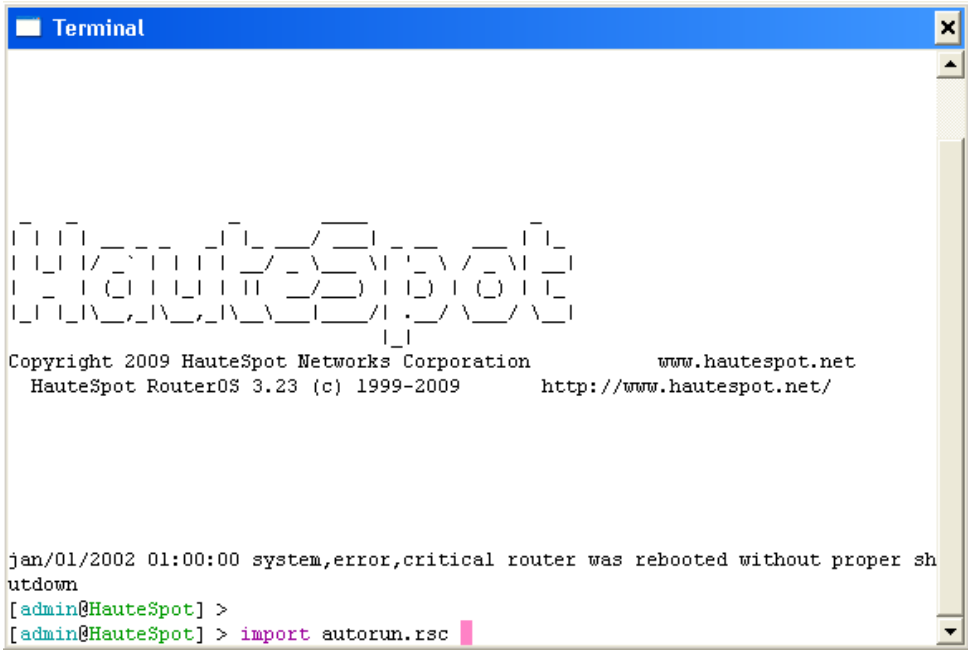
may take up to 5 minutes to complete.

35 Once the update process has finished the device will be listed in the HSConf loader. Login to the device and open the 'Files List' menu once again by selecting the 'Files' button. You can notice that the files list will look much different now that the update process has occurred. We will need to now drag in the autorun.rsc from your unzipped directory. Be sure that the file gets



dragged into the correct location (not in the hotspot/img file directory).

36 The autorun.rsc file is simply the default configuration stored in script form. To run this script open the 'New Terminal' menu in HS Configurator. To run the script use the command 'import autorun.rsc'. The device may take a minute to reboot and configure itself. If the file is not found double check that it was dragged into the correct directory.


Terminal
Copyright 2009 HauteSpot Networks Corporation www.hautespot.net
HauteSpot RouterOS 3.23 (c) 1999-2009 http://www.hautespot.net/
jan/01/2002 01:00:00 system,error,critical router was rebooted without proper sh
utdown
[admin@HauteSpot] >
[admin@HauteSpot] > import autorun.rsc

HauteRouterOS General Interface Settings

Document revision: 1.1 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

HauteRouterOS supports a variety of Network Interface Cards as well as some virtual interfaces (like Bonding, Bridge, VLAN etc.). Each of them has its own submenu, but there is also a list of all interfaces where some common properties can be configured.

Description

The Manual describes general settings of HauteRouterOS interfaces.

Interface Status

Submenu level: ***/interface***

Property Description

name (*text*) - the name of the interface
type (*read-only:* arlan | bonding | bridge | cyclades | eoip | ethernet | farsync | ipip | isdn-client | isdn-server | l2tp-client | l2tp-server | moxa-c101 | moxa-c502 | mtsync | pc | ppp-client | ppp-server | pppoe-client | pppoe-server | pptp-client | pptp-server | pvc | radiolan | sbe | vlan | wavelan | wireless | xpeed) - interface type
mtu (*integer*) - maximum transmission unit for the interface (in bytes)
rx-rate (*integer*; default: **0**) - maximum data rate for receiving data
0 - no limits
tx-rate (*integer*; default: **0**) - maximum data rate for transmitting data
0 - no limits

Example

To see the list of all available interfaces:

```
[admin@Hautespot] interface> print
```

```
Flags: X - disabled, D - dynamic, R - running
 #    NAME                          TYPE              RX-RATE    TX-RATE    MTU
 0  R ether1                        ether             0          0          1500
 1  R bridge1                       bridge            0          0          1500
 2  R ether2                        ether             0          0          1500
 3  R wlan1                         wlan              0          0          1500
[admin@Hautespot] interface>
```

Traffic Monitoring

Command name: ***/interface monitor-traffic***

Description

The traffic passing through any interface can be monitored.

Property Description

received-packets-per-second (*read-only: integer*) - number of packets that interface has received in one second
received-bits-per-second (*read-only: integer*) - number of bits that interface has received in one second
sent-packets-per-second (*read-only: integer*) - number of packets that interface has sent in one second
sent-bits-per-second (*read-only: integer*) - number of bits that interface has sent in one second

Notes

One or more interfaces can be monitored at the same time.

To see overall traffic passing through all interfaces at time, use **aggregate** instead of interface name.

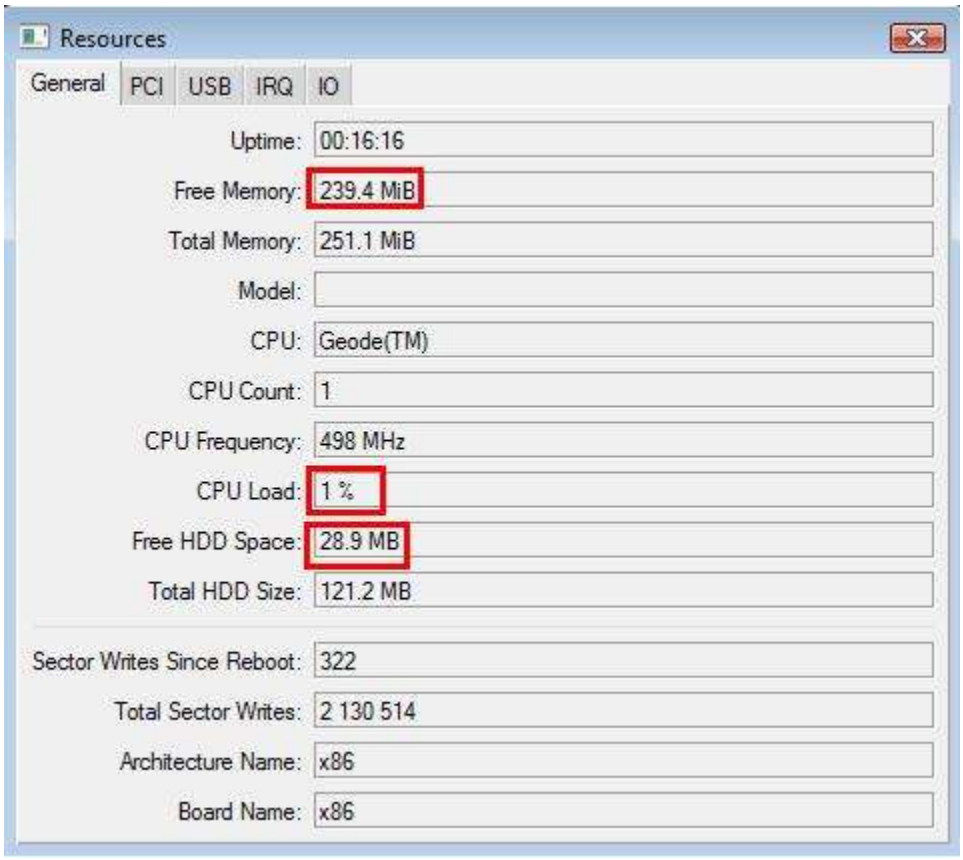
Example

Multiple interface monitoring:

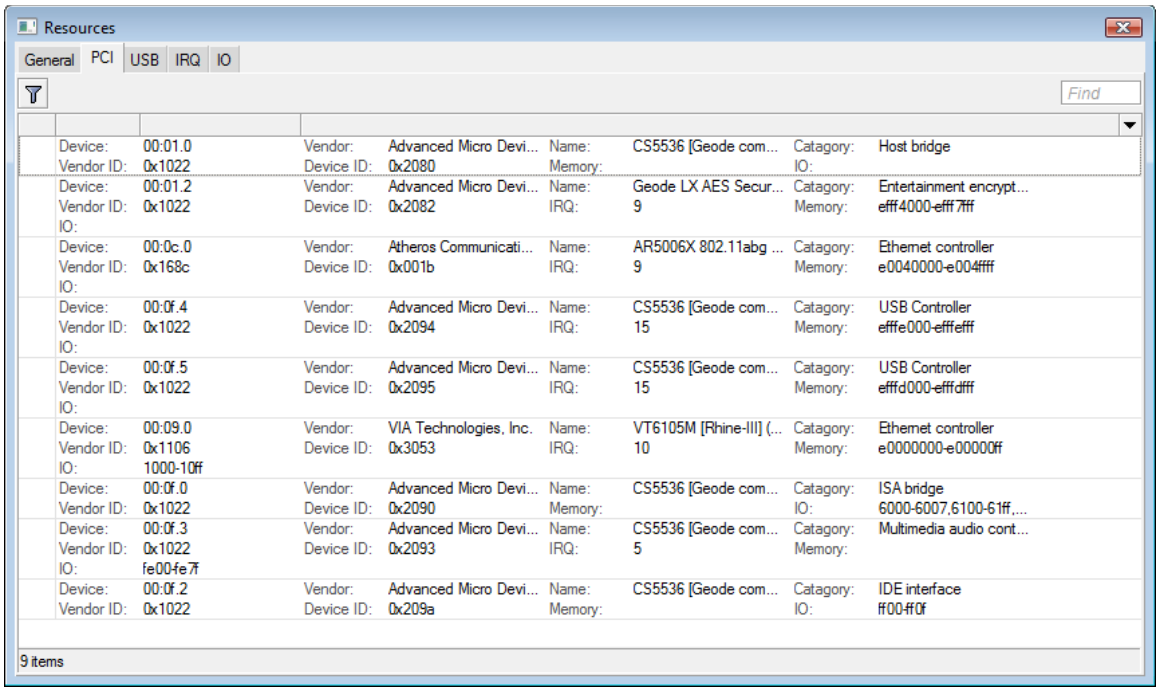
```
/interface monitor-traffic ether1,aggregate
    received-packets-per-second: 9        11
       received-bits-per-second: 4.39kbps 6.19kbps
        sent-packets-per-second: 16       17
           sent-bits-per-second: 101kbps  101kbps
-- [Q quit|D dump|C-z pause]
```

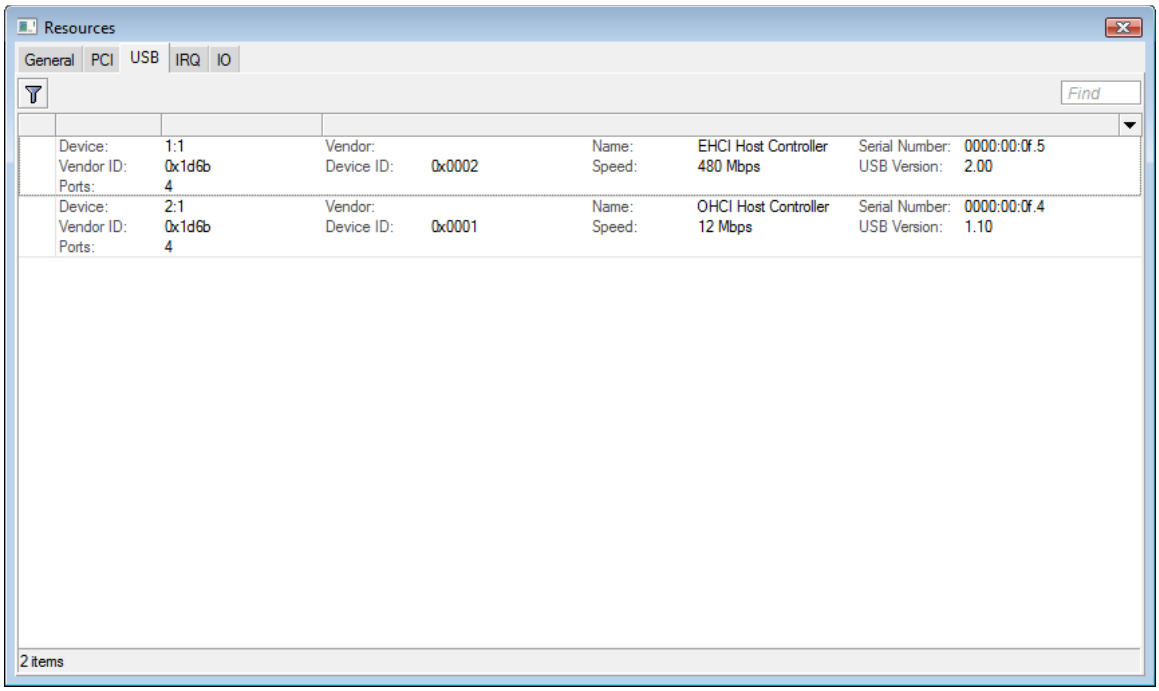
Healthy System Values for HauteWRAP HR-WRAPSX Systems

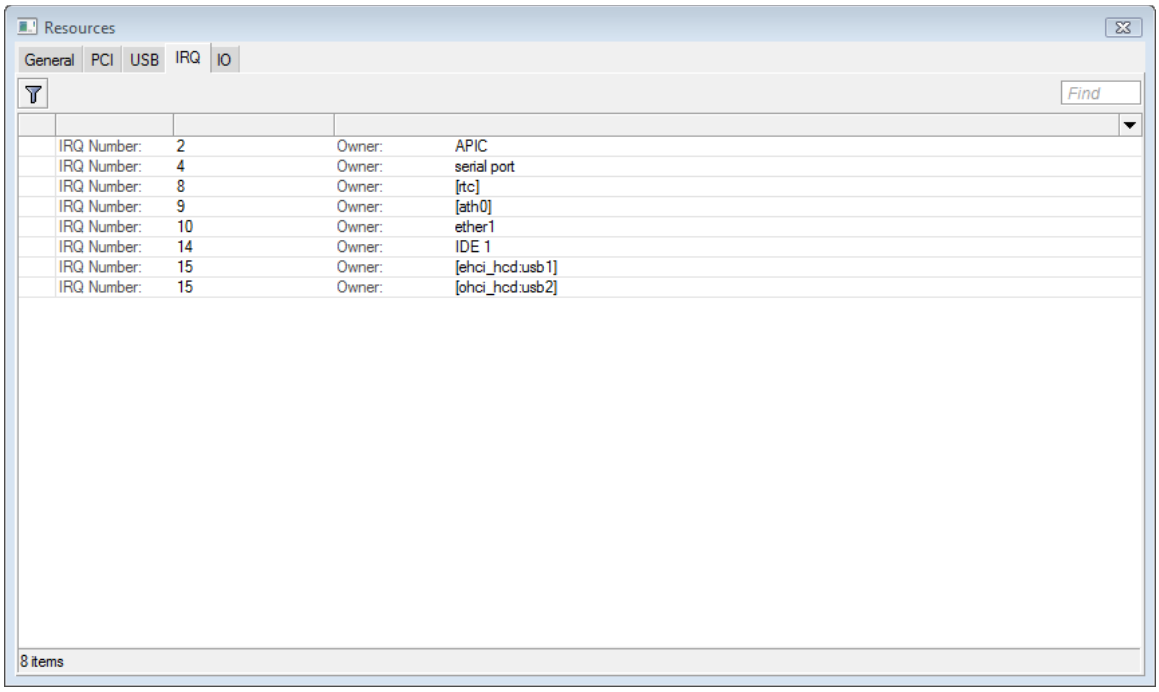
/system resources

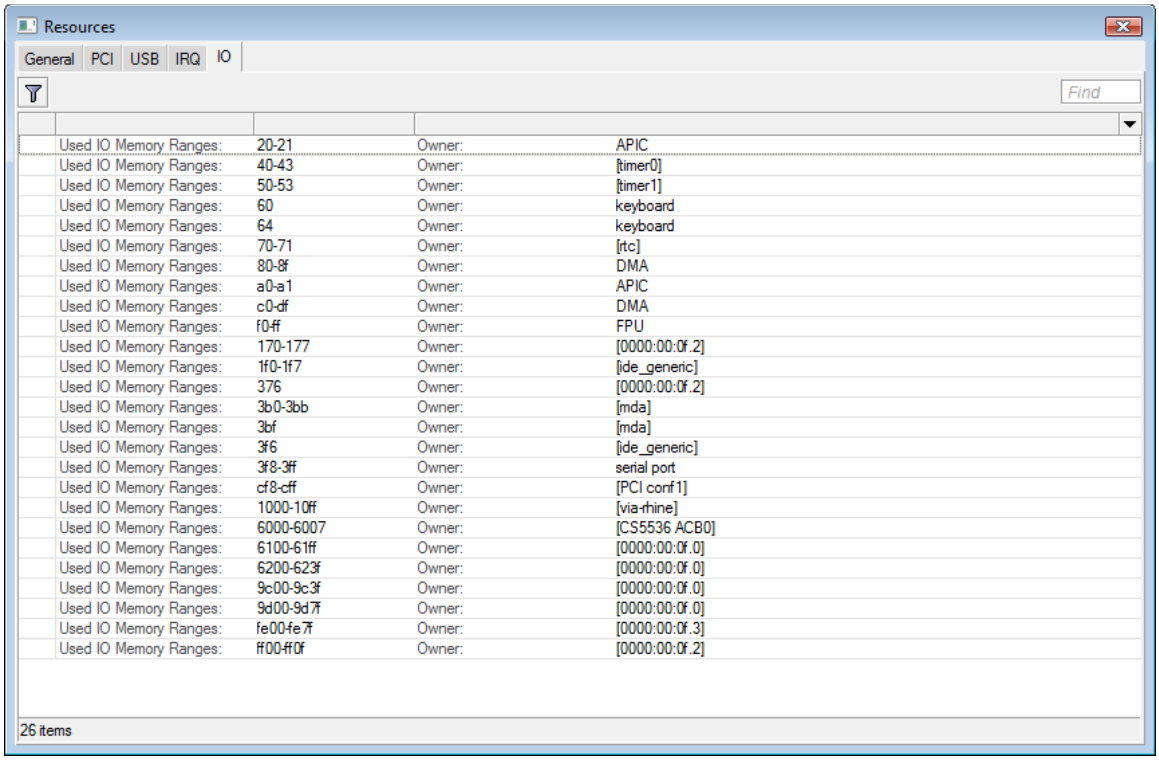


A system that has little to no free memory available may be suffering from a crash or driver lock up. Not a very common scenario. A system that has 100% CPU load is usually suffering from a bad route, bridge loop, or may just be receiving too much data from too many clients. This is pretty common and is usually solved by a simple configuration tweak. Free HDD space is required to store log data and service packages. An HDD with zero space is usually the result of an open log.

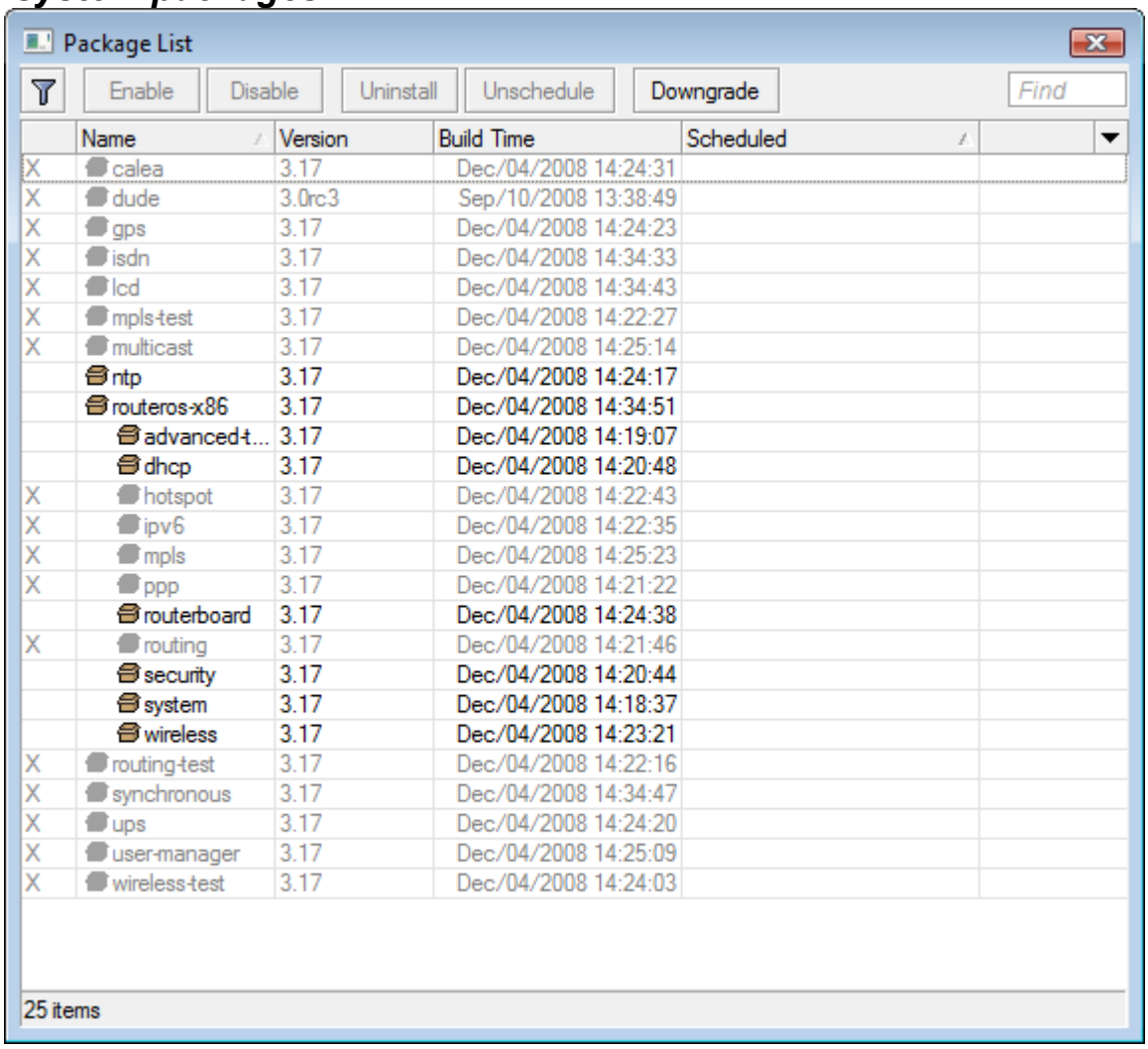

Resources
General
PCI
USB
IRQ
IO
Find
Device: 00:01.0 Vendor: Advanced Micro Devi... Name: CS5536 [Geode com... Catagory: Host bridge
Vendor ID: 0x1022 Device ID: 0x2080 Memory: IO:
Device: 00:01.2 Vendor: Advanced Micro Devi... Name: Geode LX AES Secur... Catagory: Entertainment encrypt...
Vendor ID: 0x1022 Device ID: 0x2082 IRQ: 9 Memory: efff4000-efff7fff
IO:
Device: 00:0c.0 Vendor: Atheros Communicati... Name: AR5006X 802.11abg ... Catagory: Ethernet controller
Vendor ID: 0x168c Device ID: 0x001b IRQ: 9 Memory: e0040000-e004ffff
IO:
Device: 00:0f.4 Vendor: Advanced Micro Devi... Name: CS5536 [Geode com... Catagory: USB Controller
Vendor ID: 0x1022 Device ID: 0x2094 IRQ: 15 Memory: efffe000-efffefff
IO:
Device: 00:0f.5 Vendor: Advanced Micro Devi... Name: CS5536 [Geode com... Catagory: USB Controller
Vendor ID: 0x1022 Device ID: 0x2095 IRQ: 15 Memory: efffd000-efffdfff
IO:
Device: 00:09.0 Vendor: VIA Technologies, Inc. Name: VT6105M [Rhine-III] (... Catagory: Ethernet controller
Vendor ID: 0x1106 Device ID: 0x3053 IRQ: 10 Memory: e0000000-e00000ff
IO: 1000-10ff
Device: 00:0f.0 Vendor: Advanced Micro Devi... Name: CS5536 [Geode com... Catagory: ISA bridge
Vendor ID: 0x1022 Device ID: 0x2090 Memory: IO: 6000-6007,6100-61ff,...
Device: 00:0f.3 Vendor: Advanced Micro Devi... Name: CS5536 [Geode com... Catagory: Multimedia audio cont...
Vendor ID: 0x1022 Device ID: 0x2093 IRQ: 5 Memory:
IO: fe00-fe7f
Device: 00:0f.2 Vendor: Advanced Micro Devi... Name: CS5536 [Geode com... Catagory: IDE interface
Vendor ID: 0x1022 Device ID: 0x209a Memory: IO: ff00-ff0f
9 items


Resources
General
PCI
USB
IRQ
IO
Find
Device: 1:1 Vendor: Name: EHCI Host Controller Serial Number: 0000:00:0f.5
Vendor ID: 0x1d6b Device ID: 0x0002 Speed: 480 Mbps USB Version: 2.00
Ports: 4
Device: 2:1 Vendor: Name: OHCI Host Controller Serial Number: 0000:00:0f.4
Vendor ID: 0x1d6b Device ID: 0x0001 Speed: 12 Mbps USB Version: 1.10
Ports: 4
2 items


Resources
General
PCI
USB
IRQ
IO
Find
IRQ Number: 2 Owner: APIC
IRQ Number: 4 Owner: serial port
IRQ Number: 8 Owner: [rtc]
IRQ Number: 9 Owner: [ath0]
IRQ Number: 10 Owner: ether1
IRQ Number: 14 Owner: IDE 1
IRQ Number: 15 Owner: [ehci_hcd:usb1]
IRQ Number: 15 Owner: [ohci_hcd:usb2]
8 items


Resources
General
PCI
USB
IRQ
IO
Find
Used IO Memory Ranges: 20-21 Owner: APIC
Used IO Memory Ranges: 40-43 Owner: [timer0]
Used IO Memory Ranges: 50-53 Owner: [timer1]
Used IO Memory Ranges: 60 Owner: keyboard
Used IO Memory Ranges: 64 Owner: keyboard
Used IO Memory Ranges: 70-71 Owner: [rtc]
Used IO Memory Ranges: 80-8f Owner: DMA
Used IO Memory Ranges: a0-a1 Owner: APIC
Used IO Memory Ranges: c0-df Owner: DMA
Used IO Memory Ranges: f0-ff Owner: FPU
Used IO Memory Ranges: 170-177 Owner: [0000:00:0f.2]
Used IO Memory Ranges: 1f0-1f7 Owner: [ide_generic]
Used IO Memory Ranges: 376 Owner: [0000:00:0f.2]
Used IO Memory Ranges: 3b0-3bb Owner: [mda]
Used IO Memory Ranges: 3bf Owner: [mda]
Used IO Memory Ranges: 3f6 Owner: [ide_generic]
Used IO Memory Ranges: 3f8-3ff Owner: serial port
Used IO Memory Ranges: cf8-cff Owner: [PCI conf1]
Used IO Memory Ranges: 1000-10ff Owner: [via-rhine]
Used IO Memory Ranges: 6000-6007 Owner: [CS5536 ACB0]
Used IO Memory Ranges: 6100-61ff Owner: [0000:00:0f.0]
Used IO Memory Ranges: 6200-623f Owner: [0000:00:0f.0]
Used IO Memory Ranges: 9c00-9c3f Owner: [0000:00:0f.0]
Used IO Memory Ranges: 9d00-9d7f Owner: [0000:00:0f.0]
Used IO Memory Ranges: fe00-fe7f Owner: [0000:00:0f.3]
Used IO Memory Ranges: ff00-ff0f Owner: [0000:00:0f.2]
26 items

/system packages



By default your HR-WRAP comes with several unused packages. Double check to make the active packages listed above are running on your device. You may not have some of the unused packages listed above.

HauteRouterOS HotSpot Gateway

Document revision: 4 (Tuesday, October 30, 2007)

Applies to: HauteRouterOS V3.0

General Information

Summary

The Hautespot HotSpot Gateway enables providing of public network access for clients using wireless or wired network connections.

HotSpot Gateway features:

- authentication of clients using local client database, or RADIUS server
- accounting using local database, or RADIUS server
- Walled-garden system (accessing some web pages without authorization)

Quick Setup Guide

The most noticeable difference in user experience setting up HotSpot system in version 2.9 from the previous RouterOS versions is that it has become in order of magnitude easier to set up a correctly working HotSpot system.

Given a router with two interfaces: Local (where HotSpot clients are connected to) and Public, which is connected to the Internet. To set up HotSpot on the Local interface:

1. first, a valid IP config is required on both interfaces. This can be done with **/setup** command. In this example we will assume the configuration with DHCP server on the Local interface
2. valid DNS configuration must be set up in the **/ip dns** submenu
3. To put HotSpot on the Local interface, using the same IP address pool as DHCP server uses for that interface: `/ip hotspot add interface=local address-pool=dhcp-pool-1`
4. and finally, add at least one HotSpot user: `/ip hotspot user add name=admin`

These simple steps should be sufficient to enable HotSpot system

Please find many HotSpot How-to's, which will answer most of your questions about configuring a HotSpot gateway, at the end of this manual. It is still recommended that you read and understand all the **Description** section below before deploying a HotSpot system.

If this does not work:

- check that **/ip dns** contains valid DNS servers, try to **/ping www.Hautespot.com** to see, that DNS resolving works
- make sure that connection tracking is enabled: `/ip firewall connection tracking set enabled=yes`

Specifications

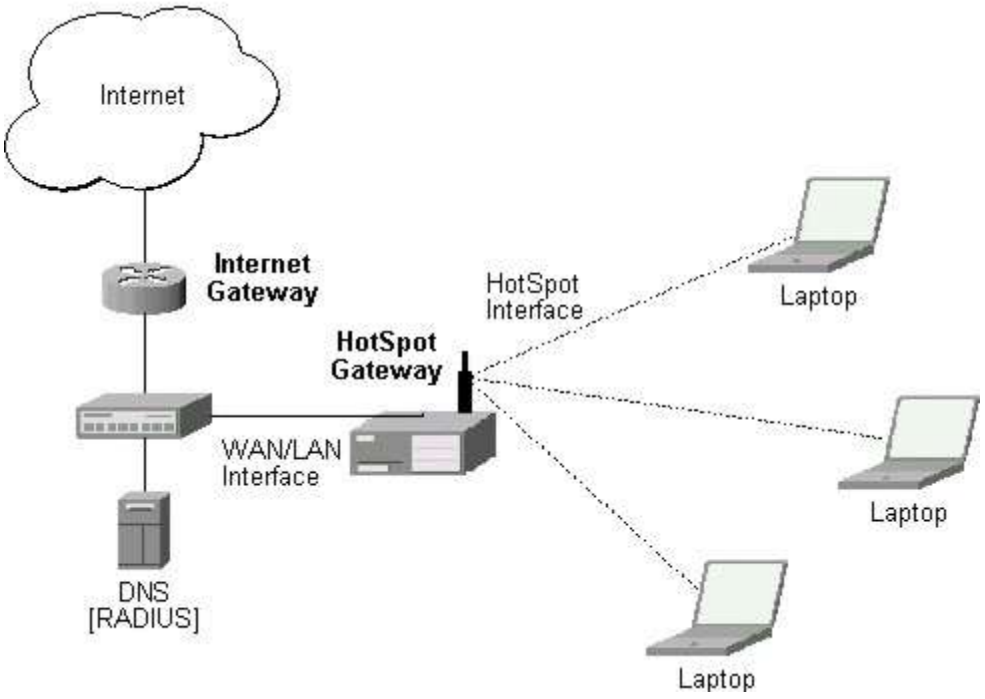
Packages required: ***hotspot***, ***dhcp****(optional)*
License required: *Level1 (Limited to 1 active user) , Level3 (Limited to 1 active user) , Level4 (Limited to 200 active users) , Level5 (Limited to 500 active users) , Level6*
Submenu level: ***/ip hotspot***
Standards and Technologies: ICMP, DHCP
Hardware usage: *Not significant*

Description

Hautespot HotSpot Gateway should have at least two network interfaces:

1. HotSpot interface, which is used to connect HotSpot clients
2. LAN/WAN interface, which is used to access network resources. For example, DNS and RADIUS server(s) should be accessible

The diagram below shows a sample HotSpot setup.



The HotSpot interface should have an IP address assigned to it. Physical network connection has to be established between the HotSpot user's computer and the gateway. It can be wireless (the wireless card should be registered to AP), or wired (the NIC card should be connected to a hub or a switch).

Note that the most noticeable difference in user experience setting up HotSpot system in version 2.9 from the previous RouterOS versions is that it has become in order of magnitude easier to set up a correctly working HotSpot system.

Introduction to HotSpot

HotSpot is a way to authorize users to access some network resources. It does not provide traffic encryption. To log in, users may use almost any web browser (either HTTP or HTTPS protocol), so they are not required to install additional software. The gateway is accounting the uptime and amount of traffic each of its clients have used, and also can send this information to a RADIUS server. The HotSpot system may limit each particular user's bitrate, total amount of traffic, uptime and some other parameters mentioned further in this document.

The HotSpot system is targeted to provide authentication within a local network (to access the Internet), but may as well be used to authorize access from outer networks to access local resources. Configuring Walled Garden feature, it is possible to allow users to access some web pages without the need of prior authentication.

Getting Address

First of all, a client must get an IP address. It may be set on the client statically, or leased from a DHCP server. The DHCP server may provide ways of binding lent IP addresses to clients MAC addresses, if required. The HotSpot system does not care how did a client get an address before he/she gets to the HotSpot login page.

Moreover, HotSpot server may automatically and transparently change any IP address (yes, meaning really **any** IP address) of a client to a valid unused address from the selected IP pool. This feature gives a possibility to provide a network access (for example, Internet access) to mobile clients that are not willing (or are disallowed, not qualified enough or otherwise unable) to change their networking settings. The users will not notice the translation (i.e., there will not be any changes in the users' config), but the router itself will see completely different (from what is actually set on each client) source IP addresses on packets sent from the clients (even firewall mangle table will 'see' the translated addresses). This technique is called one-to-one NAT, but is also known as "Universal Client" as that is how it was called in the RouterOS version 2.8.

One-to-one NAT accepts any incoming address from a connected network interface and performs a network address translation so that data may be routed through standard IP networks. Clients may use any preconfigured addresses. If the one-to-one NAT feature is set to translate a client's address to a public IP address, then the client may even run a server or any other service that requires a public IP address. This NAT is changing source address of each packet just after it is received by the router (it is like source NAT that is performed earlier, so that even firewall mangle table, which normally 'sees' received packets unaltered, can only 'see' the translated address).

Note also that **arp** mode must be **enabled** on the interface you use one-to-one NAT on.

Before the authentication

When enabling HotSpot on an interface, the system automatically sets up everything needed to show login page for all clients that are not logged in. This is done by adding dynamic destination NAT rules, which you can observe on a working HotSpot system. These rules are needed to redirect all HTTP and HTTPS requests from unauthorized users to the HotSpot servlet (i.e., the authentication procedure, e.g., the login page). Other rules that are also inserted, we will describe later in a special section of this manual.

In most common setup, opening any HTTP page will bring up the HotSpot servlet login page (which can be customized extensively, as will be described later on). As normal user behavior is to open web pages by their DNS names, a valid DNS configuration should be set up on the HotSpot gateway itself (it is possible to reconfigure the gateway so that it will not require local DNS configuration, but such a configuration is impractical and thus not recommended).

Walled Garden

You may wish not to require authorization for some services (for example to let clients access the web server of your company without registration), or even to require authorization only to a number of services (for example, for users to be allowed to access an internal file server or another restricted area). This can be done by setting up Walled Garden system.

When a not logged-in user requests a service allowed in the Walled Garden configuration, the HotSpot gateway does not intercept it, or in case of HTTP, simply redirects the request to the original destination (or to a specified parent proxy). When a user is logged in, there is no effect of this table on him/her.

To implement the Walled Garden feature for HTTP requests, an embedded web proxy server has been designed, so all the requests from not authorized users are really going through this proxy. **Note** that the embedded proxy server does not have caching function yet. Also note that this embedded proxy server is in the **system** software package and does not require **web-proxy** package. It is configurable under **/ip proxy**

Authentication

There are currently 5 different authentication methods. You can use one or more of them simultaneously:

- **HTTP PAP** - simplest method, which shows the HotSpot login page and expect to get the authentication info (i.e. username and password) in plain text. **Note** that passwords are not being encrypted when transferred over the network. An another use of this method is the possibility of hard-coded authentication information in the servlet's login page simply creating the appropriate link.
- **HTTP CHAP** - standard method, which includes CHAP challenge in the login page. The CHAP MD5 hash challenge is to be used together with the user's password for computing the string which will be sent to the HotSpot gateway.

The hash result (as a password) together with username is sent over network to HotSpot service (so, password is never sent in plain text over IP network). On the client side, MD5 algorithm is implemented in JavaScript applet, so if a browser does not support JavaScript (like, for example, Internet Explorer 2.0 or some PDA browsers), it will not be able to authenticate users. It is possible to allow unencrypted passwords to be accepted by turning on HTTP PAP authentication method, but it is not recommended (because of security considerations) to use that feature.

- **HTTPS** - the same as HTTP PAP, but using SSL protocol for encrypting transmissions. HotSpot user just send his/her password without additional hashing (note that there is no need to worry about plain-text password exposure over the network, as the transmission itself is encrypted). In either case, HTTP POST method (if not possible, then - HTTP GET method) is used to send data to the HotSpot gateway.
- **HTTP cookie** - after each successful login, a cookie is sent to web browser and the same cookie is added to active HTTP cookie list. Next time the same user will try to log in, web browser will send http cookie. This cookie will be compared with the one stored on the HotSpot gateway and only if source MAC address and randomly generated ID match the ones stored on the gateway, user will be automatically logged in using the login information (username and password pair) was used when the cookie was first generated. Otherwise, the user will be prompted to log in, and in the case authentication is successful, old cookie will be removed from the local HotSpot active cookie list and the new one with different random ID and expiration time will be added to the list and sent to the web browser. It is also possible to erase cookie on user manual logoff (not in the default server pages). This method may only be used together with HTTP PAP, HTTP CHAP or HTTPS methods as there would be nothing to generate cookies in the first place otherwise.
- **MAC address** - try to authenticate clients as soon as they appear in the hosts list (i.e., as soon as they have sent any packet to the HotSpot server), using client's MAC address as username

HotSpot can authenticate users consulting the local user database or a RADIUS server (local database is consulted first, then - a RADIUS server). In case of HTTP cookie authentication via RADIUS server, the router will send the same information to the server as was used when the cookie was first generated. If authentication is done locally, profile corresponding to that user is used, otherwise (in case RADIUS reply did not contain the group for that user) the default profile is used to set default values for parameters, which are not set in RADIUS access-accept message. For more information on how the interaction with a RADIUS server works, see the respective manual section.

The HTTP PAP method also makes it possible to authenticate by requesting the page `/login?username=username&password=password` . In case you want to log in using telnet connection, the exact HTTP request would look like that: **GET /login?username=username&password=password HTTP/1.0** (note that the request is case-sensitive)

Authorization

After authentication, user gets access to the Internet, and receives some limitations (which are user profile specific). HotSpot may also perform a one-to-one NAT for the client, so that a particular user would always receive the same IP address regardless of what PC is he/she working at.

The system will automatically detect and redirect requests to a proxy server a client is using (if any; it may be set in his/her settings to use an unknown to us proxy server) to the proxy server embedded in the router.

Authorization may be delegated to a RADIUS server, which delivers similar configuration options as the local database. For any user requiring authorization, a RADIUS server gets queried first, and if no reply received, the local database is examined. RADIUS server may send a Change of Authorization request according to standards to alter the previously accepted parameters.

Advertisement

The same proxy used for unauthorized clients to provide Walled-Garden facility, may also be used for authorized users to show them advertisement popups. Transparent proxy for authorized users allows to monitor http requests of the clients and to take some action if required. It enables the possibility to open status page even if client is logged in by mac address, as well as to show advertisements time after time

When time has come to show an advertisement, the server redirects client's web browser to the status page. Only requests, which provide html content, are redirected (images and other content will not be affected). The status page displays the advertisement and next advertise-interval is used to schedule next advertisement. If status page is unable to display an advertisement for configured timeout starting from moment, when it is scheduled to be shown, client access is blocked within walled-garden (as unauthorized clients are). Client is unblocked when the scheduled page is finally shown. Note that if popup windows are blocked in the browser, the link on the status page may be used to open the advertisement manually.

While client is blocked, FTP and other services will not be allowed. Thus requiring client to open an advertisement for any Internet activity not especially allowed by the Walled-Garden.

Accounting

The HotSpot system implement accounting internally, you are not required to do anything special for it to work. The accounting information for each user may be sent to a RADIUS server.

Configuration menus

- **/ip hotspot** - HotSpot servers on particular interfaces (one server per interface). HotSpot server must be added in this menu in order for HotSpot system to work on an interface

- **/ip hotspot profile** - HotSpot server profiles. Settings, which affect login procedure for HotSpot clients are configured here. More than one HotSpot servers may use the same profile
- **/ip hotspot host** - dynamic list of active network hosts on all HotSpot interfaces. Here you can also find IP address bindings of the one-to-one NAT
- **/ip hotspot ip-binding** - rules for binding IP addresses to hosts on hotspot interfaces
- **/ip hotspot service-port** - address translation helpers for the one-to-one NAT
- **/ip hotspot walled-garden** - Walled Garden rules at HTTP level (DNS names, HTTP request substrings)
- **/ip hotspot walled-garden ip** - Walled Garden rules at IP level (IP addresses, IP protocols)
- **/ip hotspot user** - local HotSpot system users
- **/ip hotspot user profile** - local HotSpot system users profiles (user groups)
- **/ip hotspot active** - dynamic list of all authenticated HotSpot users
- **/ip hotspot cookie** - dynamic list of all valid HTTP cookies

Question&Answer-Based Setup

Command name: ***/ip hotspot setup***

Questions

address pool of network (*name*) - IP address pool for the HotSpot network
dns name (*text*) - DNS domain name of the HotSpot gateway (will be statically configured on the local DNS proxy
dns servers (*IP address*,[*IP address*]) - DNS servers for HotSpot clients
hotspot interface (*name*) - interface to run HotSpot on
ip address of smtp server (*IP address*; default: **0.0.0.0**) - IP address of the SMTP server to redirect SMTP requests (TCP port 25) to
- **0.0.0.0** - no redirect

local address of network (*IP address*; default: **10.5.50.1/24**) - HotSpot gateway address for the interface
masquerade network (yes | no; default: **yes**) - whether to masquerade the HotSpot network
name of local hotspot user (*text*; default: **admin**) - username of one automatically created user
passphrase (*text*) - the **passphrase** of the certificate you are importing
password for the user (*text*) - password for the automatically created user
select certificate (*name* | none import-other-certificate) - choose SSL certificate from the list of the imported certificates
- **none** - do not use SSL
- **import-other-certificate** - setup the certificates not imported yet, and ask this question again

Notes

Depending on current settings and answers to the previous questions, default values of following questions may be different. Some questions may disappear if they become redundant

Example

To configure HotSpot on ether1 interface (which is already configured with address of 192.0.2.1/25), and adding user admin with password rubbish:

```
[admin@Hautespot] > ip hotspot setup
hotspot interface: ether1
local address of network: 192.0.2.1/24
masquerade network: yes
address pool of network: 192.0.2.2-192.0.2.126
select certificate: none
ip address of smtp server: 0.0.0.0
dns servers: 192.0.2.254
dns name: hs.example.net
name of local hotspot user: admin
password for the user: rubbish
[admin@Hautespot] >
```

HotSpot Interface Setup

Submenu level: ***/ip hotspot***

Description

HotSpot system is put on individual interfaces. You can run completely different HotSpot configurations on different interfaces

Property Description

addresses-per-mac (*integer* | unlimited; default: **2**) - number of IP addresses allowed to be bind with any particular MAC address (it is a small chance to reduce denial of service attack based on taking over all free IP addresses)
unlimited - number of IP addresses per one MAC address is not limited
address-pool (*name* | none; default: **none**) - IP address pool name for performing one-to-one NAT. You can choose not to use the one-to-one NAT
none - do not perform one-to-one NAT for the clients of this HotSpot interface
HTTPS (*read-only: flag*) - whether the HTTPS service is actually running on the interface (i.e., it is set up in the server profile, and a valid certificate is imported in the router)
idle-timeout (*time* | none; default: **00:05:00**) - idle timeout (maximal period of inactivity) for unauthorized clients. It is used to detect, that client is not using outer networks (e.g. Internet), i.e., there is NO TRAFFIC coming from that client and going

through the router. Reaching the timeout, user will be dropped of the host list, and the address used buy the user will be freed
none - do not timeout idle users
interface (*name*) - interface to run HotSpot on
ip-of-dns-name (*read-only: IP address*) - IP address of the HotSpot gateway's DNS name set in the HotSpot interface profile
keepalive-timeout (*time* | none; default: **none**) - keepalive timeout for unauthorized clients. Used to detect, that the computer of the client is alive and reachable. If check will fail during this period, user will be dropped of the host list, and the address used buy the user will be freed
none - do not timeout unreachable users
profile (*name*; default: **default**) - default HotSpot profile for the interface

Command Description

reset-html (*name*) - overwrite the existing HotSpot servlet with the original HTML files. It is used if you have changed the servlet and it is not working after that

Notes

addresses-per-mac property works only if address pool is defined. Also note that in case you are authenticating users connected through a router, than all the IP addresses will seem to have come from one MAC address.

Example

To add HotSpot system to the **local** interface, allowing the system to do one-to-one NAT for each client (addresses from the **HS-real** address pool will be used for the NAT):

```
[admin@Hautespot] ip hotspot> add interface=local address-pool=HS-real
[admin@Hautespot] ip hotspot> print
Flags: X - disabled, I - invalid, S - HTTPS
 #   NAME                      INTERFACE    ADDRESS-POOL PROFILE IDLE-
TIMEOUT
 0   hs-local                  local        HS-real      default 00:05:00
[admin@Hautespot] ip hotspot>
```

HotSpot Server Profiles

Submenu level: ***/ip hotspot profile***

Property Description

dns-name (*text*) - DNS name of the HotSpot server. This is the DNS name used as the name of the HotSpot server (i.e., it appears as the location of the login page). This name will automatically be added as a static DNS entry in the DNS cache
hotspot-address (*IP address*; default: **0.0.0.0**) - IP address for HotSpot service
html-directory (*text*; default: **""**) - name of the directory (accessible with FTP), which stores the HTML servlet pages (when changed, the default pages are automatically copied into specified directory if it does not exist already)
http-cookie-lifetime (*time*; default: **3d**) - validity time of HTTP cookies
http-proxy (*IP address*; default: **0.0.0.0**) - the address of the proxy server the HotSpot service will use as a proxy server for all those requests intercepted by Universal Proxy system and not defined in the **/ip proxy direct** list. If not specified, the address defined in **parent-proxy** parameter of **/ip proxy**. If that is absent too, the request will be resolved by the local proxy
login-by (*multiple choice:* cookie | http-chap | http-pap | https | mac | trial; default: **cookie,http-chap**) - which authentication methods to use
cookie - use HTTP cookies to authenticate, without asking user credentials. Other method will be used in case the client does not have cookie, or the stored username and password pair are not valid anymore since the last authentication. May only be used together with other HTTP authentication methods (HTTP-PAP, HTTP-CHAP or HTTPS), as in the other case there would be no way for the cookies to be generated in the first place
http-chap - use CHAP challenge-response method with MD5 hashing algorithm for hashing passwords. This way it is possible to avoid sending clear-text passwords over an insecure network. This is the default authentication method
http-pap - use plain-text authentication over the network. Please note that in case this method will be used, your user passwords will be exposed on the local networks, so it will be possible to intercept them
https - use encrypted SSL tunnel to transfer user communications with the HotSpot server. Note that in order this to work, a valid certificate must be imported into the router (see a separate manual on certificate management)
mac - try to use client's MAC address first as its username. If the matching MAC address exists in the local user database or on the RADIUS server, the client will be authenticated without asking to fill the login form
trial - does not require authentication for a certain amount of time
radius-accounting (yes | no; default: **yes**) - whether to send RADIUS server accounting information on each user once in a while (the "while" is defined in the **radius-interim-update** property)
radius-interim-update (*time* | received; default: **received**) - how often to sent cumulative accounting reports.
0s - same as **received**
received - use whatever value received from the RADIUS server
rate-limit (*text*; default: **""**) - Rate limitation in form of **rx-rate[/tx-rate] [rx-burst-rate[/tx-burst-rate] [rx-burst-threshold[/tx-burst-threshold] [rx-burst-time[/tx-burst-time]]]]** from the point of view of the router (so "rx" is client upload, and "tx" is client download). All rates should be numbers with optional 'k' (1,000s) or 'M' (1,000,000s). If tx-rate is not specified, rx-rate is as tx-rate too. Same goes for tx-burst-rate and tx-burst-threshold and tx-burst-time. If both rx-burst-threshold and tx-burst-threshold are not specified (but burst-rate is specified), rx-rate and tx-rate is used as burst thresholds. If both rx-burst-time and tx-burst-time are not specified, 1s is used as default
smtp-server (*IP address*; default: **0.0.0.0**) - default SMTP server to be used to redirect unconditionally all user SMTP requests to
split-user-domain (yes | no; default: **no**) - whether to split username from domain name when the username is given in "user@domain" or in "domain\user" format

ssl-certificate (*name* | none; default: **none**) - name of the SSL certificate to use for HTTPS authentication. Not used for other authentication methods
trial-uptime (*time*/*time*; default: **30m/1d**) - is used only when authentication method is trial. Specifies the amount of time the user identified by MAC address can use hotspot services without authentication and the time, that has to pass that the user is allowed to use hotspot services again
trial-user-profile (*name*; default: **default**) - is used only only when authentication method is trial. Specifies user profile, that trial users will use
use-radius (yes | no; default: **no**) - whether to use RADIUS to authenticate HotSpot users

Notes

If **dns-name** property is not specified, **hotspot-address** is used instead. If **hotspot-address** is also absent, then both are to be detected automatically.

In order to use RADIUS authentication, the **/radius** menu must be set up accordingly.

Trial authentication method should allways be used together with one of the other authentication methods.

Example

HotSpot User Profiles

Submenu level: ***/ip hotspot user profile***

Description

Article moved to: HotSpot AAA section

HotSpot Users

Submenu level: ***/ip hotspot user***

Description

Article moved to: HotSpot AAA section

HotSpot Active Users

Submenu level: ***/ip hotspot active***

Description

Article moved to: HotSpot AAA section

HotSpot Cookies

Submenu level: ***/ip hotspot cookie***

Description

Cookies can be used for authentication in the Hotspot service

Property Description

domain (*read-only: text*) - domain name (if split from username)
expires-in (*read-only: time*) - how long the cookie is valid
mac-address (*read-only: MAC address*) - user's MAC address
user (*read-only: name*) - username

Notes

There can be multiple cookies with the same MAC address. For example, there will be a separate cookie for each web browser on the same computer.

Cookies can expire - that's the way how it is supposed to be. Default validity time for cookies is **3** days (72 hours), but it can be changed for each individual HotSpot server profile, for example :

```
/ip hotspot profile set default http-cookie-lifetime=1d
```

Example

To get the list of valid cookies:

```
[admin@Hautespot] ip hotspot cookie> print
  # USER               DOMAIN            MAC-ADDRESS       EXPIRES-IN
  0 ex                                   01:23:45:67:89:AB 23h54m16s
[admin@Hautespot] ip hotspot cookie>
```

HTTP-level Walled Garden

Submenu level: ***/ip hotspot walled-garden***

Description

Walled garden is a system which allows unauthorized use of some resources, but requires authorization to access other resources. This is useful, for example, to give access to some general information about HotSpot service provider or billing options.

This menu only manages Walled Garden for HTTP and HTTPS protocols. Other protocols can also be included in Walled Garden, but that is configured elsewhere (in **/ip hotspot walled-garden ip**; see the next section of this manual for details)

Property Description

action (allow | deny; default: **allow**) - action to undertake if a packet matches the rule:
allow - allow the access to the page without prior authorization
deny - the authorization is required to access this page
dst-address (*IP address*) - IP address of the destination web server
dst-host (*wildcard*; default: **""**) - domain name of the destination web server (this is a wildcard)
dst-port (*integer*; default: **""**) - the TCP port a client has send the request to
method (*text*) - HTTP method of the request
path (*text*; default: **""**) - the path of the request (this is a wildcard)
server (*name*) - name of the HotSpot server this rule applied to
src-address (*IP address*) - IP address of the user sending the request

Notes

Wildcard properties (**dst-host** and **dst-path**) match a complete string (i.e., they will not match "example.com" if they are set to "example"). Available wildcards are '*' (match any number of any characters) and '?' (match any one character). Regular expressions are also accepted here, but if the property should be treated as a regular expression, it should start with a colon (':').

Small hits in using regular expressions:

- **** symbol sequence is used to enter **** character in console
- **\.** pattern means **.** only (in regular expressions single dot in pattern means any symbol)
- to show that no symbols are allowed before the given pattern, we use **^** symbol at the beginning of the pattern
- to specify that no symbols are allowed after the given pattern, we use **$** symbol at the end of the pattern

You can not use **path** property for HTTPS requests as router can not (and should not - that is what the HTTPS protocol was made for!) decrypt the request.

Example

To allow unauthorized requests to the **www.example.com** domain's **/paynow.html** page:

```
[admin@Hautespot] ip hotspot walled-garden> add path="/paynow.html" \
\... dst-host="www.example.com"
[admin@Hautespot] ip hotspot walled-garden> print
Flags: X - disabled, D - dynamic
 0   dst-host="www.example.com" path="/paynow.html" action=allow
[admin@Hautespot] ip hotspot walled-garden>
```

IP-level Walled Garden

Submenu level: ***/ip hotspot walled-garden ip***

Description

This menu is manages Walled Garden for generic IP requests. See the previous section for managing HTTP and HTTPS protocol specific properties (like the actual DNS name, HTTP method and path used in requests).

Property Description

action (accept | drop | reject; default: **accept**) - action to undertake if a packet matches the rule:
accept - allow the access to the page without prior authorization
drop - the authorization is required to access this page
reject - the authorization is required to access this page, in case the page will be accsessed withot authorization ICMP reject message host-unreachable will be generated
dst-address (*IP address*) - IP address of the destination web server
dst-host (*text*; default: **""**) - domain name of the destination web server (this is **not** a regular expression or a wildcard of any kind). The DNS name specified is resolved to a list of IP addresses when the rule is added, and all those IP addresses are used
dst-port (*integer*; default: **""**) - the TCP or UDP port (protocol **MUST** be specified explicitly in the **protocol** property) a client has send the request to
protocol (*integer* | ddp egp encap ggp gre hmp icmp idpr-cmtp igmp ipencap ipip ipsec-ah ipsec-esp iso-tp4 ospf pup rdp rspf st tcp udp vmtp xns-idp xtp) - IP protocol name
server (*name*) - name of the HotSpot server this rule applied to
src-address (*IP address*) - IP address of the user sending the request

Example

One-to-one NAT static address bindings

Submenu level: ***/ip hotspot ip-binding***

Description

You can setup NAT translations statically based on either the original IP address (or IP network), or the original MAC address. You can also allow some addresses to bypass HotSpot authentication (i.e., they will be able work without having to log in to the network first) and completely block some addresses.

Property Description

address (*IP address* / [*netmask*]; default: **""**) - the original IP address or network of the client
mac-address (*MAC address*; default: **""**) - the source MAC address of the client
server (*name*|all; default: **all**) - the name of the server the client is connecting to
to-address (*IP address*; default: **""**) - IP address to translate the original client address to. If **address** property is given as network, this is the starting address for the translation (i.e., the first **address** is translated to **to-address**, **address** + 1 to **to-address** + 1, and so on)
type (regular | bypassed | blocked) - type of the static binding entry
regular - perform a one-to-one NAT translation according to the values set in this entry
bypassed - perform the translation, but exclude the client from having to log in to the HotSpot system
blocked - the translation will not be preformed, and all packets from the host will be dropped

Notes

This is an ordered list, so you can put more specific entries on the top of the list for them to override the more common that appear lower.

Active Host List

Submenu level: ***/ip hotspot host***

Description

This menu shows all active network hosts that are connected to the HotSpot gateway. This list includes all one-to-one NAT translations

Property Description

address (*read-only: IP address*) - the original IP address of the client
authorized (*read-only: flag*) - whether the client is successfully authenticated by the HotSpot system
blocked (*read-only: flag*) - true, if access is blocked within walled-garden because of expired advertisement timeout
bridge-port (*read-only: name*) - the actual physical interface, which the host is connected to. This is used when HotSpot service is put on a bridge interface to determine the host's actual port within the bridge.

bypass-hotspot (*read-only: flag*) - whether the client does not need to be authorized by the HotSpot system
bytes-in (*read-only: integer*) - how many bytes did the router receive from the client
bytes-out (*read-only: integer*) - how many bytes did the router send to the client
host-dead-time (*read-only: time*) - how long has the router not received any packets (including ARP replies, keepalive replies and user traffic) from this host
idle-time (*read-only: time*) - the amount of time has the user been idle
idle-timeout (*read-only: time*) - the exact value of **idle-timeout** that applies to this user. This property shows how long should the user stay idle for it to be logged off automatically
keepalive-timeout (*read-only: time*) - the exact value of **keepalive-timeout** that applies to this user. This property shows how long should the user's computer stay out of reach for it to be logged off automatically
mac-address (*read-only: MAC address*) - the actual MAC address of the user
packets-in (*read-only: integer*) - how many packets did the router receive from the client
packets-out (*read-only: integer*) - how many packets did the router send to the client
server (*read-only: name*) - name of the server, which the host is connected to
static (*read-only: flag*) - whether this translation has been taken from the static IP binding list
to-address (*read-only: IP address*) - what address is the original IP address of the host translated to
uptime (*read-only: time*) - current session time of the user (i.e., how long has the user been in the active host list)

Command Description

make-binding - copy a dynamic entry from this list to the static IP bindings list
Input Parameters
unnamed (*name*) - item number
comment (*text*) - custom comment to the static entry to be created
type (regular | bypassed | blocked) - the type of the static entry

Service Port

Submenu level: ***/ip hotspot service-port***

Description

Just like for classic NAT, the HotSpot embedded one-to-one NAT 'breaks' some protocols that are incompatible with address translation. To leave these protocols consistent, helper modules must be used. For the one-to-one NAT the only such a module is for FTP protocol.

Property Description

name (*read-only: name*) - protocol name
ports (*read-only: integer*) - list of the ports on which the protocol is working

Example

To set the FTP protocol uses both 20 and 21 TCP port:

```
[admin@Hautespot] ip hotspot service-port> print
Flags: X - disabled
  #   NAME
PORTS
  0   ftp                                                                 21
[admin@Hautespot] ip hotspot service-port> set ftp ports=20,21
[admin@Hautespot] ip hotspot service-port> print
Flags: X - disabled
  #   NAME
PORTS
  0   ftp                                                                 20
                                                                          21
[admin@Hautespot] ip hotspot service-port>
```

Customizing HotSpot: Firewall Section

Description

Apart from the obvious dynamic entries in the **/ip hotspot** submenu itself (like hosts and active users), some additional rules are added in the firewall tables when activating a HotSpot service. Unlike RouterOS version 2.8, there are relatively few firewall rules added in the firewall as the main job is made by the one-to-one NAT algorithm.

NAT rules

From **/ip firewall nat print dynamic** command, you can get something like this (comments follow after each of the rules):

```
 0 D chain=dstnat hotspot=from-client action=jump jump-target=hotspot
```

Putting all HotSpot-related tasks for packets from all HotSpot clients into a separate chain

```
 1 D chain=hotspot protocol=udp dst-port=53 action=redirect to-ports=64872
 2 D chain=hotspot protocol=tcp dst-port=53 action=redirect to-ports=64872
```

Redirect all DNS requests to the HotSpot service. The 64872 port provides DNS service for all HotSpot users. If you want HotSpot server to listen also to another port, add rules here the same way, changing **dst-port** property

```
 3 D chain=hotspot protocol=tcp dst-port=80 hotspot=local-dst action=redirect
     to-ports=64873
```

Redirect all HTTP login requests to the HTTP login servlet. The 64873 is HotSpot HTTP servlet port.

```
 4 D chain=hotspot protocol=tcp dst-port=443 hotspot=local-dst action=redirect
     to-ports=64875
```

Redirect all HTTPS login requests to the HTTPS login servlet. The 64875 is HotSpot HTTPS servlet port.

```
 5 D chain=hotspot protocol=tcp action=jump hotspot=!auth jump-target=hs-
unauth
```

All other packets except DNS and login requests from unauthorized clients should pass through the **hs-unauth** chain

```
 6 D chain=hotspot protocol=tcp action=jump hotspot=auth jump-target=hs-auth
```

And packets from the authorized clients - through the **hs-auth** chain

```
 7 D ;;; www.Hautespot.com
     chain=hs-unauth dst-address=159.148.147.196 protocol=tcp dst-port=80
     action=return
```

First in the **hs-unauth** chain is put everything that affects TCP protocol in the **/ip hotspot walled-garden ip** submenu (i.e., everything where either protocol is not set, or set to TCP). Here we are excluding www.Hautespot.com from being redirected to the login page.

```
 8 D chain=hs-unauth protocol=tcp dst-port=80 action=redirect to-ports=64874
```

All other HTTP requests are redirected to the Walled Garden proxy server which listens the 64874 port. If there is an **allow** entry in the **/ip hotspot walled-garden** menu for an HTTP request, it is being forwarded to the destination. Otherwise, the request will be automatically redirected to the HotSpot login servlet (port 64873).

```
 9 D chain=hs-unauth protocol=tcp dst-port=3128 action=redirect to-ports=64874
10 D chain=hs-unauth protocol=tcp dst-port=8080 action=redirect to-ports=64874
```

HotSpot by default assumes that only these ports may be used for HTTP proxy requests. These two entries are used to "catch" client requests to unknown proxies. I.e., to make it possible for the clients with unknown proxy settings to work with the HotSpot system. This feature is called "Universal Proxy". If it is detected that a client is using some proxy server, the system will automatically mark that packets with the **http** hotspot mark to work around the unknown proxy problem, as we will see later on. Note that the port used (64874) is the same as for HTTP requests in the rule #8 (so both HTTP and HTTP proxy requests are processed by the same code).

```
11 D chain=hs-unauth protocol=tcp dst-port=443 action=redirect to-ports=64875
```

HTTPS proxy is listening on the 64875 port

```
12 D chain=hs-unauth protocol=tcp dst-port=25 action=jump jump-target=hs-smtp
```

Redirect for SMTP protocol may also be defined in the HotSpot configuration. In case it is, a redirect rule will be put in the **hs-smtp** chain. This is done so that users with unknown SMTP configuration would be able to send their mail through the service provider's (your) SMTP server instead of going to [possibly unavailable outside their network of origin] the SMTP server users have configured in their computers.

```
13 D chain=hs-auth protocol=tcp hotspot=http action=redirect to-ports=64874
```

Providing HTTP proxy service for authorized users. Authenticated user requests may need to be subject to the transparent proxying (the "Universal Proxy" technique and for the advertisement feature). This **http** mark is put automatically on the HTTP proxy requests to the servers detected by the HotSpot HTTP proxy (the one that is listening on the 64874 port) to be HTTP proxy requests to unknown proxy servers. This is done so that users that have some proxy settings would use the HotSpot gateway instead of the [possibly unavailable outside their network of origin] proxy server users have configured in their computers. The mark is as well put on any HTTP requests done form the users whoose profile is configured to transparently proxy their requests.

```
14 D chain=hs-auth protocol=tcp dst-port=25 action=jump jump-target=hs-smtp
```

Providing SMTP proxy for authorized users (the same as in rule #12)

Packet filter rules

From **/ip firewall filter print dynamic** command, you can get something like this (comments follow after each of the rules):

```
 0 D chain=forward hotspot=from-client,!auth action=jump jump-target=hs-unauth
```

Any packet that traverse the router from unauthorized client will be sent to the **hs-unauth** chain. The **hs-unauth** implements the IP-based Walled Garden filter.

```
 1 D chain=forward hotspot=to-client,!auth action=jump jump-target=hs-unauth-
to
```

Everything that comes to clients through the router, gets redirected to another chain, called **hs-unauth-to**. This chain should reject unauthorized requests to the clients

```
 2 D chain=input hotspot=from-client action=jump jump-target=hs-input
```

Everything that comes from clients to the router itself, gets to another chain, called **hs-input**.

```
 3 D chain=hs-input protocol=udp dst-port=64872 action=accept
 4 D chain=hs-input protocol=tcp dst-port=64872-64875 action=accept
```

Allow client access to the local authentication and proxy services (as described earlier)

```
 5 D chain=hs-input hotspot=!auth action=jump jump-target=hs-unauth
```

All other traffic from unauthorized clients to the router itself will be treated the same way as the traffic traversing the routers

```
 6 D chain=hs-unauth protocol=icmp action=return
 7 D ;;; www.Hautespot.com
     chain=hs-unauth dst-address=159.148.147.196 protocol=tcp dst-port=80
     action=return
```

Unlike NAT table where only TCP-protocol related Walled Garden entries were added, in the packet filter **hs-unauth** chain is added everything you have set in the **/ip hotspot walled-garden ip** menu. That is why although you have seen only one entry in the NAT table, there are two rules here.

```
 8 D chain=hs-unauth protocol=tcp action=reject reject-with=tcp-reset
 9 D chain=hs-unauth action=reject reject-with=icmp-net-prohibited
```

Everything else that has not been while-listed by the Walled Garden will be rejected. Note usage of TCP Reset for rejecting TCP connections.

```
10 D chain=hs-unauth-to action=reject reject-with=icmp-host-prohibited
```

Reject all packets to the clients with ICMP reject message

Customizing HotSpot: HTTP Servlet Pages

Description

You can create a completely different set of servlet pages for each HotSpot server you have, specifying the directory it will be stored in **html-directory** property of a HotSpot server profile (**/ip hotspot profile**). The default servlet pages are copied in the directory of your choice right after you create the profile. This directory can be accessed by connecting to the router with an FTP client. You can modify the pages as you like using the information from this section of the manual.

Available Servlet Pages

Main HTML servlet pages, which are shown to user:

- **redirect.html** - redirects user to another url (for example, to login page)
- **login.html** - login page shown to a user to ask for username and password
- **md5.js** - javascript for md5 password encryption. Used together with **http-chap** login method
- **alogin.html** - page shown after client has logged in. It pops-up status page and redirects browser to originally requested page (before he/she was redirected to the HotSpot login page)

- **status.html** - status page, shows statistics for the client
- **logout.html** - logout page, shown after user is logged out. Shows final statistics about the finished session
- **error.html** - error page, shown on fatal errors only

Some other pages are available as well, if more control is needed:

- **rlogin.html** - page, which redirects client from some other URL to the login page, if authorization of the client is required to access that URL
- **rstatus.html** - similarly to rlogin.html, only in case if the client is already logged in and the original URL is not known
- **flogin.html** - shown instead of login.html, if some error has happened (invalid username or password, for example)
- **fstatus.html** - shown instead of redirect, if status page is requested, but client is not logged in
- **flogout.html** - shown instead of redirect, if logout page is requested, but client is not logged in

Serving Servlet Pages

The HotSpot servlet recognizes 5 different request types:

1. request for a remote host
    - if user is logged in, the requested page is served
    - if user is not logged in, but the destination host is allowed by walled garden, then the request is also served
    - if user is not logged in, and the destination host is disallowed by walled garden, **rlogin.html** is displayed; if **rlogin.html** is not found, **redirect.html** is used to redirect to the login page
2. request for "/" on the HotSpot host
    - if user is logged in, **rstatus.html** is displayed; if **rstatus.html** is not found, **redirect.html** is used to redirect to the status page
    - if user is not logged in, **rlogin.html** is displayed; if **rlogin.html** is not found, **redirect.html** is used to redirect to the login page
3. request for "/login" page
    - if user has successfully logged in (or is already logged in), **alogin.html** is displayed; if **alogin.html** is not found, **redirect.html** is used to redirect to the originally requested page or the status page (in case, original destination page was not given)
    - if user is not logged in (username was not supplied, no error message appeared), **login.html** is showed
    - if login procedure has failed (error message is supplied), **flogin.html** is displayed; if **flogin.html** is not found, **login.html** is used
    - in case of fatal errors, **error.html** is showed
4. request for "/status" page
    - if user is logged in, **status.html** is displayed
    - if user is not logged in, **fstatus.html** is displayed; if **fstatus.html** is not found, **redirect.html** is used to redirect to the login page
5. request for '/logout' page
    - if user is logged in, **logout.html** is displayed

    - if user is not logged in, **flogout.html** is displayed; if **flogout.html** is not found, **redirect.html** is used to redirect to the login page

Note that if it is not possible to meet a request using the pages stored on the router's FTP server, Error 404 is displayed

There are many possibilities to customize what the HotSpot authentication pages look like:

- The pages are easily modifiable. They are stored on the router's FTP server in the directory you choose for the respective HotSpot server profile.
- By changing the variables, which client sends to the HotSpot servlet, it is possible to reduce keyword count to one (username or password; for example, the client's MAC address may be used as the other value) or even to zero (License Agreement; some predefined values general for all users or client's MAC address may be used as username and password)
- Registration may occur on a different server (for example, on a server that is able to charge Credit Cards). Client's MAC address may be passed to it, so that this information need not be written in manually. After the registration, the server may change RADIUS database enabling client to log in for some amount of time.

To insert variable in some place in HTML file, the $(var_name) syntax is used, where the "var_name" is the name of the variable (without quotes). This construction may be used in any HotSpot HTML file accessed as '/', '/login', '/status' or '/logout', as well as any text or HTML file stored on the HotSpot server. For example, to show a link to the login page, following construction can be used:

```
<a href="$(link-login)">login</a>
```

Variables

All of the Servlet HTML pages use variables to show user specific values. Variable names appear only in the HTML source of the servlet pages - they are automatically replaced with the respective values by the HotSpot Servlet. For each variable there is an example of its possible value included in brackets. All the described variables are valid in all servlet pages, but some of them just might be empty at the time they are accesses (for example, there is no uptime before a user has logged in).

- Common server variables:
    - **hostname** - DNS name or IP address (if DNS name is not given) of the HotSpot Servlet ("hotspot.example.net")
    - **identity** - RouterOS identity name ("Hautespot")
    - **login-by** - authentication method used by user
    - **plain-passwd** - a "yes/no" representation of whether HTTP-PAP login method is allowed ("no")
    - **server-address** - HotSpot server address ("10.5.50.1:80")
    - **ssl-login** - a "yes/no" representation of whether HTTPS method was used to access that servlet page ("no")

- **trial** - a "yes/no" representation of whether the user has access to trial time. If users trial time has expired, the value is "no"

Links:

- **link-login** - link to login page including original URL requested ("http://10.5.50.1/login?dst=http://www.example.com/")
- **link-login-plain** - link to login page, not including original URL requested ("http://10.5.50.1/login")
- **link-logout** - link to logout page ("http://10.5.50.1/logout")
- **link-status** - link to status page ("http://10.5.50.1/status")
- **link-orig** - original URL requested ("http://www.example.com/")

General client information

- **domain** - domain name of the user ("mt.lv")
- **ip** - IP address of the client ("10.5.50.2")
- **logged-in** - "yes" if the user is logged in, otherwise - "no" ("yes")
- **mac** - MAC address of the user ("01:23:45:67:89:AB")
- **username** - the name of the user ("John")

User status information:

- **idle-timeout** - idle timeout ("20m" or "---" if none)
- **idle-timeout-secs** - idle timeout in seconds ("88" or "" if there is such timeout)
- **limit-bytes-in** - byte limit for send ("1000000" or "---" if there is no limit)
- **limit-bytes-out** - byte limit for receive ("1000000" or "---" if there is no limit)
- **refresh-timeout** - status page refresh timeout ("1m30s")
- **refresh-timeout-secs** - status page refresh timeout in seconds ("90s")
- **session-timeout** - session time left for the user ("5h" or "---" if none)
- **session-timeout-secs** - session time left for the user, in seconds ("3475" or "" if there is such timeout)
- **uptime** - current session uptime ("10h2m33s")
- **uptime-secs** - current session uptime in seconds ("125")

Traffic counters, which are available only in status page:

- **bytes-in** - number of bytes received from the user ("15423")
- **bytes-in-nice** - user-friendly form of number of bytes received from the user ("15423")
- **bytes-out** - number of bytes sent to the user ("11352")
- **bytes-out-nice** - user-friendly form of number of bytes sent to the user ("11352")
- **packets-in** - number of packets received from the user ("251")
- **packets-out** - number of packets sent to the user ("211")

- **remain-bytes-in** - remaining bytes until limit-bytes-in will be reached ("337465" or "---" if there is no limit)
- **remain-bytes-out** - remaining bytes until limit-bytes-out will be reached ("124455" or "---" if there is no limit)

Miscellaneous variables

- **session-id** - value of 'session-id' parameter in the last request
- **var** - value of 'var' parameter in the last request
- **error** - error message, if something failed ("invalid username or password")
- **error-orig** - original error message (without translations retrieved from **errors.txt**), if something failed ("invalid username or password")
- **chap-id** - value of chap ID ("\371")
- **chap-challenge** - value of chap challenge ("\357\015\330\013\021\234\145\245\303\253\142\246\133\175\375\316")
- **popup** - whether to pop-up checkbox ("true" or "false")
- **advert-pending** - whether an advertisement is pending to be displayed ("yes" or "no")

Working with variables

`$(if <var_name>)` statements can be used in theses pages. Following content will be included, if value of <var_name> will not be an empty string. It is an equivalent to `$(if <var_name> != "")` It is possible to compare on equivalence as well: `$(if <var_name> == <value>)` These statements have effect until `$(elif <var_name>)`, `$(else)` or `$(endif)`. In general case it looks like this:

```
some content, which will always be displayed
$(if username == john)
Hey, your username is john
$(elif username == dizzy)
Hello, Dizzy! How are you? Your administrator.
$(elif ip == 10.1.2.3)
You are sitting at that crappy computer, which is damn slow...
$(elif mac == 00:01:02:03:04:05)
This is an ethernet card, which was stolen few months ago...
$(else)
I don't know who you are, so lets live in peace.
$(endif)
other content, which will always be displayed
```

Only one of those expressions will be shown. Which one - depends on values of those variables for each client.

Customizing Error Messages

All error messages are stored in the **errors.txt** file within the respective HotSpot servlet directory. You can change and translate all these messages to your native language. To do so, edit the **errors.txt** file. You can also use variables in the messages. All instructions are given in that file.

Multiple Versions of HotSpot Pages

Multiple hotspot page sets for the same hotspot server are supported. They can be chosen by user (to select language) or automatically by JavaScript (to select PDA/regular version of HTML pages).

To utilize this feature, create subdirectories in HotSpot HTML directory, and place those HTML files, which are different, in that subdirectory. For example, to translate everything in Latvian, subdirectory "lv" can be created with login.html, logout.html, status.html, alogin.html, radvert.html and errors.txt files, which are translated into Latvian. If the requested HTML page can not be found in the requested subdirectory, the corresponding HTML file from the main directory will be used. Then main login.html file would contain link to "/lv/login?dst=$(link-orig-esc)", which then displays Latvian version of login page: `<a href="/lv/login?dst=$(link-orig-esc)">Latviski</a>` . And Latvian version would contain link to English version: `<a href="/login?dst=$(link-orig-esc)">English</a>`

Another way of referencing directories is to specify 'target' variable:

```
        <a href="$(link-login-only)?dst=$(link-orig-
esc)&target=lv">Latviski</a>
        <a href="$(link-login-only)?dst=$(link-orig-
esc)&target=%2F">English</a>
```

After preferred directory has been selected (for example, "lv"), all links to local HotSpot pages will contain that path (for example, `$(link-status) = "http://hotspot.mt.lv/lv/status"`). So, if all hotspot pages reference links using "$(link-xxx)" variables, then no more changes are to be made - each client will stay within the selected directory all the time.

Notes

If you want to use HTTP-CHAP authentication method it is supposed that you include the **doLogin()** function (which references to the **md5.js** which must be already loaded) before the **Submit** action of the login form. Otherwise, CHAP login will fail.

The resulting password to be sent to the HotSpot gateway in case of HTTP-CHAP method, is formed MD5-hashing the concatenation of the following: chap-id, the password of the user and chap-challenge (in the given order)

In case if variables are to be used in link directly, then they must be escaped accordingly. For example, in login page, <a

href="https://login.example.com/login?mac=$(mac)&user=$(username)">link will not work as intended, if username will be "123&456=1 2". In this case instead of $(user), its escaped version must be used: $(user-esc): link. Now the same username will be converted to "123%26456%3D1+2", which is the valid representation of "123&456=1 2" in URL. This trick may be used with any variables, not only with $(username).

There is a boolean parameter "erase-cookie" to the logout page, which may be either "on" or "true" to delete user cookie on logout (so that the user would not be automatically logged on when he/she opens a browser next time.

Example

With basic HTML language knowledge and the examples below it should be easy to implement the ideas described above.

- To provide predefined value as username, in login.html change:

  ```
  <type="text" value="$(username)>
  ```

  to this line:

  ```
  <input type="hidden" name="user" value="hsuser">
  ```

  (where **hsuser** is the username you are providing)

- To provide predefined value as password, in login.html change:

  ```
  <input type="password">
  ```

  to this line:

  ```
  <input type="hidden" name="password" value="hspass">
  ```

  (where **hspass** is the password you are providing)

- To send client's MAC address to a registration server in form of:

  ```
  https://www.server.serv/register.html?mac=XX:XX:XX:XX:XX:XX
  ```

change the Login button link in login.html to:

```
https://www.server.serv/register.html?mac=$(mac)
```

(you should correct the link to point to your server)

- To show a banner after user login, in alogin.html after

```
$(if popup == 'true')
```

add the following line:

```
open('http://your.web.server/your-banner-page.html', 'my-banner-
name','');
```

(you should correct the link to point to the page you want to show)

- To choose different page shown after login, in login.html change:

```
<input type="hidden" name="dst" value="$(link-orig)">
```

to this line:

```
<input type="hidden" name="dst" value="http://your.web.server">
```

(you should correct the link to point to your server)

- To erase the cookie on logoff, in the page containing link to the logout (for example, in status.html) change:

```
open('$(link-logout)', 'hotspot_logout', ...
```

to this:

```
open('$(link-logout)?erase-cookie=on', 'hotspot_logout', ...
```

or alternatively add this line:

```
<input type="hidden" name="erase-cookie" value="on">
```

before this one:

```
<input type="submit" value="log off">
```

An another example is making HotSpot to authenticate on a remote server (which may, for example, perform creditcard charging):

- Allow direct access to the external server in walled-garden (either HTTP-based, or IP-based)
- Modify login page of the HotSpot servlet to redirect to the external authentication server. The external server should modify RADIUS database as needed

  Here is an example of such a login page to put on the HotSpot router (it is redirecting to https://auth.example.com/login.php, replace with the actual address of an external authentication server):

```
<html>
<title>...</title>
<body>
<form name="redirect" action="https://auth.example.com/login.php"
method="post">
<input type="hidden" name="mac" value="$(mac)">
<input type="hidden" name="ip" value="$(ip)">
<input type="hidden" name="user" value="$(username)">
<input type="hidden" name="link-login" value="$(link-login)">
<input type="hidden" name="link-orig" value="$(link-orig)">
<input type="hidden" name="error" value="$(error)">
</form>
<script language="JavaScript">
<!--
 document.redirect.submit();
//-->
</script>
</body>
</html>
```

- The external server can log in a HotSpot client by redirecting it back to the original HotSpot servlet login page, specifying the correct username and password

  Here is an example of such a page (it is redirecting to https://hotspot.example.com/login, replace with the actual address of a HotSpot router; also, it is displaying www.Hautespot.com after successful login, replace with what needed):

```
<html>
<title>Hotspot login page</title>
<body>
<form name="login" action="https://hotspot.example.com/login"
method="post">
<input type="text" name="username" value="demo">
<input type="password" name="password" value="none">
<input type="hidden" name="domain" value="">
<input type="hidden" name="dst" value="http://www.Hautespot.com/">
<input type="submit" name="login" value="log in">
</form>
</body>
</html>
```

- Hotspot will ask RADIUS server whether to allow the login or not. If not allowed, alogin.html page will be displayed (it can be modified to do anything!). If not allowed, flogin.html (or login.html) page will be displayed, which will redirect client back to the external authentication server.
- Note: as shown in these examples, HTTPS protocol and POST method can be used to secure communications.

Possible Error Messages

Description

There are two kinds of errors: fatal non-fatal. Fatal errors are shown on a separate HTML page called error.html. Non-fatal errors are basically indicating incorrect user actions and are shown on the login form.

General non-fatal errors:

- **You are not logged in** - trying to access the status page or log off while not logged in. **Solution**: log in
- **already authorizing, retry later** - authorization in progress. Client already has issued an authorization request which is not yet complete. **Solution**: wait for the current request to be completed, and then try again
- **chap-missing = web browser did not send challenge response (try again, enable JavaScript)** - trying to log in with HTTP-CHAP method using MD5 hash, but HotSpot server does not know the challenge used for the hash. This may happen if you use BACK buttons in browser; if JavaScript is not enabled in web browser; if **login.html** page is not valid; or if challenge value has expired on server (more than 1h of inactivity). **Solution**: instructing browser to reload (refresh) the login page usually helps if JavaScript is enabled and **login.html** page is valid
- **invalid username ($(username)): this MAC address is not yours** - trying to log in using a MAC address username different from the actual user's MAC address. **Solution**: no - users with usernames that look like a MAC

address (eg., 12:34:56:78:9a:bc) may only log in from the MAC address specified as their user name
- **session limit reached ($(error-orig))** - depending on licence number of active hotspot clients is limited to some number. The error is displayed when this limit is reached. **Solution**: try to log in later when there will be less concurrent user sessions, or buy an another license that allows more simultaneous sessions
- **hotspot service is shutting down** - RouterOS is currently being restarted or shut down. **Solution**: wait until the service will be available again

General fatal errors:

- **internal error ($(error-orig))** - this should never happen. If it will, error page will be shown displaying this error message (error-orig will describe what has happened). **Solution**: correct the error reported
- **configuration error ($(error-orig))** - the HotSpot server is not configured properly (error-orig will describe what has happened). **Solution**: correct the error reported
- **cannot assign ip address - no more free addresses from pool** - unable to get an IP address from an IP pool as there is no more free IP addresses in that pool. **Solution**: make sure there is a sufficient amount of free IP addresses in IP pool

Local HotSpot user database non-fatal errors:

- **invalid username or password** - self-explanatory
- **user $(username) is not allowed to log in from this MAC address** - trying to log in from a MAC address different from specified in user database. **Solution**: log in from the correct MAC address or take out the limitation
- **user $(username) has reached uptime limit** - self-explanatory
- **user $(username) has reached traffic limit** - either **limit-bytes-in** or **limit-bytes-out** limit is reached
- **no more sessions are allowed for user $(username)** - the **shared-users** limit for the user's profile is reached. **Solution**: wait until someone with this username logs out, use different login name or extend the **shared-users** limit

RADIUS client non-fatal errors:

- **invalid username or password** - RADIUS server has rejected the username and password sent to it without specifying a reason. **Cause**: either wrong username and/or password, or other error. **Solution**: should be clarified in RADIUS server's log files
- **<error_message_sent_by_radius_server>** - this may be any message (any text string) sent back by RADIUS server. Consult with your RADIUS server's documentation for further information

RADIUS client fatal errors:

- **RADIUS server is not responding** - user is being authenticated by RADIUS server, but no response is received from it. **Solution**: check whether the RADIUS server is running and is reachable from the HotSpot router

HotSpot How-to's

Description

This section will focus on some simple examples of how to use your HotSpot system, as well as give some useful ideas.

Setting up https authorization

At first certificate must be present with decrypted private key:

```
[admin@Hautespot] > /certificate print
Flags: K - decrypted-private-key, Q - private-key, R - rsa, D - dsa
 0 KR name="hotspot.example.net"
      subject=C=LV,L=Riga,O=MT,OU=dev,CN=hotspot.example.net,
              emailAddress=admin@hotsot.example.net
      issuer=C=LV,L=Riga,O=MT,OU=dev,CN=hotsot.example.net,
             emailAddress=admin@hotsot.example.net
      serial-number="0" email=admin@hotsot.example.net
      invalid-before=oct/27/2004 11:43:22 invalid-after=oct/27/2005 11:43:22
      ca=yes
```

Then we can use that certificate for hotspot:

```
/ip hotspot profile set default login-by=cookie,http-chap,https \
ssl-certificate=artis.hotspot.mt.lv
```

After that we can see, that HTTPS is running on hotspot interface:

```
[admin@Hautespot] > /ip hotspot print
Flags: X - disabled, I - invalid, S - HTTPS
 #   NAME                        INTERFACE    ADDRESS-POOL PROFILE IDLE-
TIMEOUT
 0 S hs-local                    local                     default 00:05:00
```

Bypass hotspot for some devices in hotspot network

All IP binding entries with **type** property set to **bypassed**, will not be asked to authorize - it means that they will have login-free access:

```
[admin@Hautespot] ip hotspot ip-binding> print
Flags: X - disabled, P - bypassed, B - blocked
 #   MAC-ADDRESS       ADDRESS         TO-ADDRESS      SERVER
 0 P                   10.11.12.3
```

If all fields has been filled in the ip-binding table and **type** has been set to **bypassed**, then the IP address of this entry will be accessible from public interfaces immediately:

```
[admin@Hautespot] ip hotspot ip-binding> print
Flags: X - disabled, P - bypassed, B - blocked
 #   MAC-ADDRESS       ADDRESS         TO-ADDRESS      SERVER
 0 P                   10.11.12.3
 1 P 00:01:02:03:04:05 10.11.12.3      10.11.12.3      hs-local
[admin@Hautespot] ip hotspot ip-binding> .. host print
Flags: S - static, H - DHCP, D - dynamic, A - authorized, P - bypassed
 #    MAC-ADDRESS       ADDRESS         TO-ADDRESS      SERVER   IDLE-TIMEOUT
 0 SB 00:01:02:03:04:05 10.11.12.3      10.11.12.3      hs-local
```

HauteRouterOS HotSpot User AAA

Document revision: 2.3 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

This document provides information on authentication, authorization and accounting parameters and configuration for HotSpot gateway system.

Specifications

Packages required: ***system***
License required: *Level1*
Submenu level: ***/ip hotspot user***
Standards and Technologies: RADIUS
Hardware usage: *Local traffic accounting requires additional memory*

Description

HotSpot User Profiles

Submenu level: ***/ip hotspot user profile***

Description

HotSpot User profiles are used for common user settings. Profiles are like user groups, they are grouping users with the same limits.

Property Description

address-pool (*name* | none; default: **none**) - the IP poll name which the users will be given IP addresses from. This works like **dhcp-pool** method in earlier versions of HauteRouterOS, except that it does not use DHCP, but rather the embedded one-to-one NAT
none - do not reassign IP addresses to the users of this profile
advertise (yes | no; default: **no**) - whether to enable forced advertisement popups for this profile
advertise-interval (*multiple choice: time*; default: **30m,10m**) - set of intervals between showing advertisement popups. After the list is done, the last value is used for all further advertisements
advertise-timeout (*time* | immediately never; default: **1m**) - how long to wait for advertisement to be shown, before blocking network access with walled-garden
advertise-url (*multiple choice: text*; default: **http://www.Hautespot.com/,http://www.HauteWRAP.com/**) - list of URLs to show as advertisement popups. The list is cyclic, so when the last item reached, next time the first is shown
idle-timeout (*time* | none; default: **none**) - idle timeout (maximal period of inactivity) for authorized clients. It is used to detect, that client is not using outer networks (e.g. Internet), i.e., there is NO TRAFFIC coming from that client and going through the router. Reaching the timeout, user will be logged out, dropped of the host list, the address used by the user will be freed, and the session time accounted will be decreased by this value
none - do not timeout idle users
incoming-filter (*name*) - name of the firewall chain applied to incoming packets from the users of this profile
incoming-packet-mark (*name*) - packet mark put on all the packets from every user of this profile automatically
keepalive-timeout (*time* | none; default: **00:02:00**) - keepalive timeout for authorized clients. Used to detect, that the computer of the client is alive and reachable. If check will fail during this period, user will be logged out, dropped of the host list, the address used by the user will be freed, and the session time accounted will be decreased by this value
none - do not timeout unreachable users
name (*name*) - profile reference name
on-login (*text*; default: **""**) - script name to launch after a user has logged in

on-logout (*text*; default: **""**) - script name to launch after a user has logged out
open-status-page (always | http-login; default: **always**) - whether to show status page also for users authenticated using mac login method. Useful if you want to put some information (for example, banners or popup windows) in the alogin.html page so that all users would see it
http-login - open status page only in case of http login (including cookie and https login methods)
always - open http status page in case of mac login as well
outgoing-filter (*name*) - name of the firewall chain applied to outgoing packets to the users of this profile
outgoing-packet-mark (*name*) - packet mark put on all the packets to every user of this profile automatically
rate-limit (*text*; default: **""**) - Rate limitation in form of **rx-rate[/tx-rate] [rx-burst-rate[/tx-burst-rate] [rx-burst-threshold[/tx-burst-threshold] [rx-burst-time[/tx-burst-time] [priority] [rx-rate-min[/tx-rate-min]]]]** from the point of view of the router (so "rx" is client upload, and "tx" is client download). All rates should be numbers with optional 'k' (1,000s) or 'M' (1,000,000s). If tx-rate is not specified, rx-rate is as tx-rate too. Same goes for tx-burst-rate and tx-burst-threshold and tx-burst-time. If both rx-burst-threshold and tx-burst-threshold are not specified (but burst-rate is specified), rx-rate and tx-rate is used as burst thresholds. If both rx-burst-time and tx-burst-time are not specified, 1s is used as default. Priority takes values 1..8, where 1 implies the highest priority, but 8 - the lowest. If rx-rate-min and tx-rate-min are not specified rx-rate and tx-rate values are used. The rx-rate-min and tx-rate-min values can not exceed rx-rate and tx-rate values.
session-timeout (*time*; default: **0s**) - session timeout (maximal allowed session time) for client. After this time, the user will be logged out unconditionally
0 - no timeout
shared-users (*integer*; default: **1**) - maximal number of simultaneously logged in users with the same username
status-autorefresh (*time* | none; default: **none**) - HotSpot servlet status page autorefresh interval
transparent-proxy (yes | no; default: **yes**) - whether to use transparent HTTP proxy for the authorized users of this profile

Notes

When idle-timeout or keepalive is reached, session-time for that user is reduced by the actual period of inactivity in order to prevent the user from being overcharged.

Example

HotSpot Users

Submenu level: ***/ip hotspot user***

Property Description

address (*IP address*; default: **0.0.0.0**) - static IP address. If not **0.0.0.0**, client will always get the same IP address. It implies, that only one simultaneous login for that user is allowed. Any existing address will be replaced with this one using the embedded one-to-one NAT
bytes-in (*read-only: integer*) - total amount of bytes received from user
bytes-out (*read-only: integer*) - total amount of bytes sent to user
limit-bytes-in (*integer*; default: **0**) - maximum amount of bytes user can transmit (i.e., bytes received from the user)
0 - no limit
limit-bytes-out (*integer*; default: **0**) - maximum amount of bytes user can receive (i.e., bytes sent to the user)
0 - no limit
limit-uptime (*time*; default: **0s**) - total uptime limit for user (pre-paid time)
0s - no limit
mac-address (*MAC address*; default: **00:00:00:00:00:00**) - static MAC address. If not **00:00:00:00:00:00**, client is allowed to login only from that MAC address
name (*name*) - user name. If authentication method is trial, then user name will be set automaticly after following pattern "T-MAC_adress", where MAC_address is trial user Mac address
packets-in (*read-only: integer*) - total amount of packets received from user (i.e., packets received from the user)
packets-out (*read-only: integer*) - total amount of packets sent to user (i.e., packets sent to the user)
password (*text*) - user password
profile (*name*; default: **default**) - user profile
routes (*text*) - routes that are to be registered on the HotSpot gateway when the client is connected. The route format is: "dst-address gateway metric" (for example, "10.1.0.0/24 10.0.0.1 1"). Several routes may be specified separated with commas
server (*name* | all; default: **all**) - which server is this user allowed to log in to
uptime (*read-only: time*) - total time user has been logged in

Notes

In case of **mac** authentication method, clients' MAC addresses can be used as usernames (without password)

The byte limits are total limits for each user (not for each session as at **/ip hotspot active**). So, if a user has already downloaded something, then session limit will show the total limit - (minus) already downloaded. For example, if download limit for a user is 100MB and the user has already downloaded 30MB, then session download limit after login at **/ip hotspot active** will be 100MB - 30MB = 70MB.

Should a user reach his/her limits (bytes-in >= limit-bytes-in or bytes-out >= limit-bytes-out), he/she will not be able to log in anymore.

The statistics is updated if a user is authenticated via local user database each time he/she logs out. It means, that if a user is currently logged in, then the statistics will not show current total values. Use **/ip hotspot active** submenu to view the statistics on the current user sessions.

If the user has IP address specified, only one simultaneous login is allowed. If the same credentials are used again when the user is still active, the active one will be automatically logged off.

Example

To add user **ex** with password **ex** that is allowed to log in only with **01:23:45:67:89:AB** MAC address and is limited to 1 hour of work:

```
[admin@Hautespot] ip hotspot user> add name=ex password=ex \
\... mac-address=01:23:45:67:89:AB limit-uptime=1h
[admin@Hautespot] ip hotspot user> print
Flags: X - disabled
 #   SERVER     NAME                      ADDRESS         PROFILE UPTIME
 0              ex                                        default 00:00:00
[admin@Hautespot] ip hotspot user> print detail
Flags: X - disabled
  0    name="ex" password="ex" mac-address=01:23:45:67:89:AB profile=default
     limit-uptime=01:00:00 uptime=00:00:00 bytes-in=0 bytes-out=0
     packets-in=0 packets-out=0
[admin@Hautespot] ip hotspot user>
```

HotSpot Active Users

Submenu level: ***/ip hotspot active***

Description

The active user list shows the list of currently logged in users. Nothing can be changed here, except user can be logged out with the **remove** command

Property Description

address (*read-only: IP address*) - IP address of the user
blocked (*read-only: flag*) - whether the user is blocked by advertisement (i.e., usual due advertisement is pending)
bytes-in (*read-only: integer*) - how many bytes did the router receive from the client
bytes-out (*read-only: integer*) - how many bytes did the router send to the client
domain (*read-only: text*) - domain of the user (if split from username)
idle-time (*read-only: time*) - the amount of time has the user been idle
idle-timeout (*read-only: time*) - the exact value of **idle-timeout** that applies to this user. This property shows how long should the user stay idle for it to be logged off automatically
keepalive-timeout (*read-only: time*) - the exact value of **keepalive-timeout** that applies to this user. This property shows how long should the user's computer stay out of reach for it to be logged off automatically

limit-bytes-in (*read-only: integer*) - maximal amount of bytes the user is allowed to send to the router
limit-bytes-out (*read-only: integer*) - maximal amount of bytes the router is allowed to send to the client
login-by (*multiple choice, read-only:* cookie | http-chap | http-pap | https | mac | trial) - authentication method used by user
mac-address (*read-only: MAC address*) - actual MAC address of the user
packets-in (*read-only: integer*) - how many packets did the router receive from the client
packets-out (*read-only: integer*) - how many packets did the router send to the client
radius (*read-only:* yes | no) - whether the user was authenticated via RADIUS
server (*read-only: name*) - the particular server the used is logged on at.
session-time-left (*read-only: time*) - the exact value of **session-time-left** that applies to this user. This property shows how long should the user stay logged-in (see **uptime**) for it to be logged off automatically
uptime (*read-only: time*) - current session time of the user (i.e., how long has the user been logged in)
user (*read-only: name*) - name of the user

Example

To get the list of active users:

```
[admin@Hautespot] ip hotspot active> print
Flags: R - radius, B - blocked
 #    USER            ADDRESS         UPTIME       SESSION-TIMEOUT IDLE-
TIMEOUT
 0    ex              10.0.0.144      4m17s        55m43s
[admin@Hautespot] ip hotspot active>
```

HauteRouterOS HSconf

Document revision: 1.0 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

The HauteRouterOS can be configured remotely, using **Telnet**, **SSH**, **HSconf Console** or **Webcfg**. In this manual we will discuss how to use the interactive **HSconf** console.

Description

The HSconf console is used for accessing the HauteSpot Router configuration and management features, using graphical user interface (GUI).

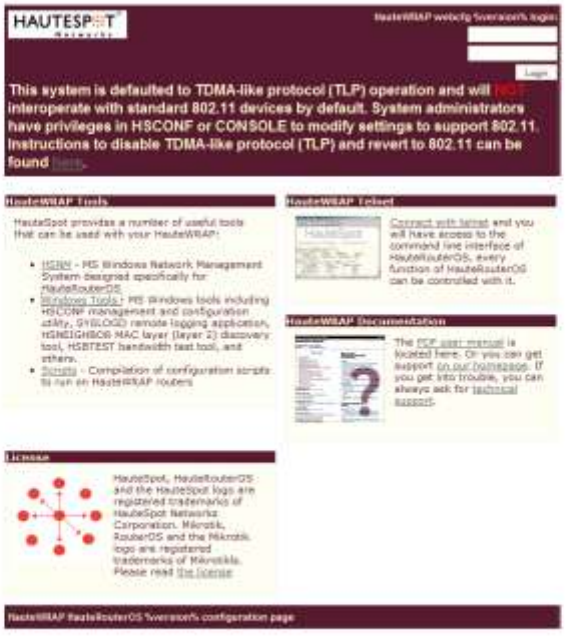
All HSconf interface functions are as close as possible to Console functions: all HSconf functions are exactly in the same hierarchy in Terminal Console and vice versa (except functions that are not implemented in HSconf). That is why there are no HSconf sections in the manual.

The HSconf Console plugin loader, the HSconf.exe program, can be retrieved from the HauteSpot router, the URL is **http://router_address/HSconf/HSconf.exe** Use any web browser on Windows 95/98/ME/NT4.0/2000/XP or Linux to retrieve the **HSconf.exe** executable file from Router. If your router is not specifically configured, you can also type in the web-browser just **http://router_address**

The HSconf plugins are cached on the local disk for each HauteRouterOS version. The plugins are not downloaded, if they are in the cache, and the router has not been upgraded since the last time it has been accessed.

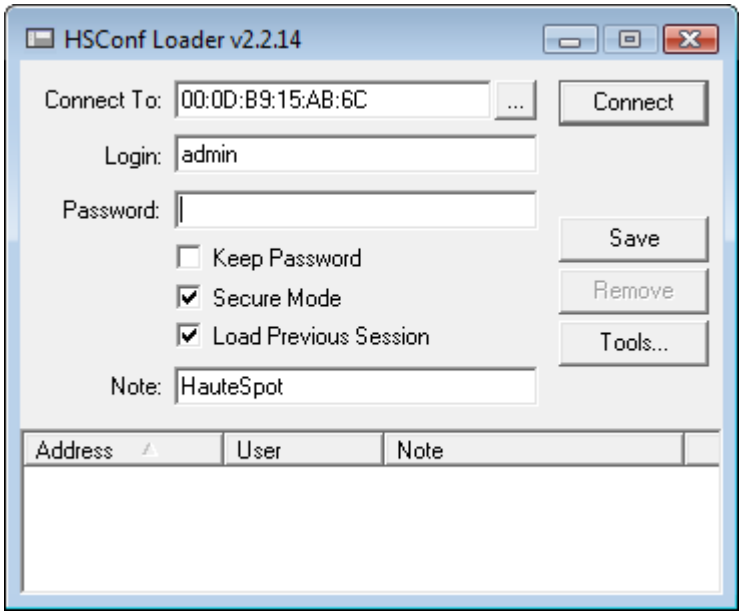
Starting the HSconf Console

When connecting to the HauteSpot router via http (TCP port 80 by default), the router's Welcome Page is displayed in the web browser:


HAUTESPOT
This system is defaulted to TDMA-like protocol (TLP) operation and will
interoperate with standard 802.11 devices by default. System administrators
have privileges in HSCONF or CONSOLE to modify settings to support 802.11.
Instructions to disable TDMA-like protocol (TLP) and revert to 802.11 can be
found
HauteWRAP Tools
HauteWRAP Telnet
HauteWRAP Documentation
License

By clicking on the HSconf link you can start the HSconf.exe download. Choose **Open** to start the HSconf loader program (you can also save this program to your local disk, and run it from there)

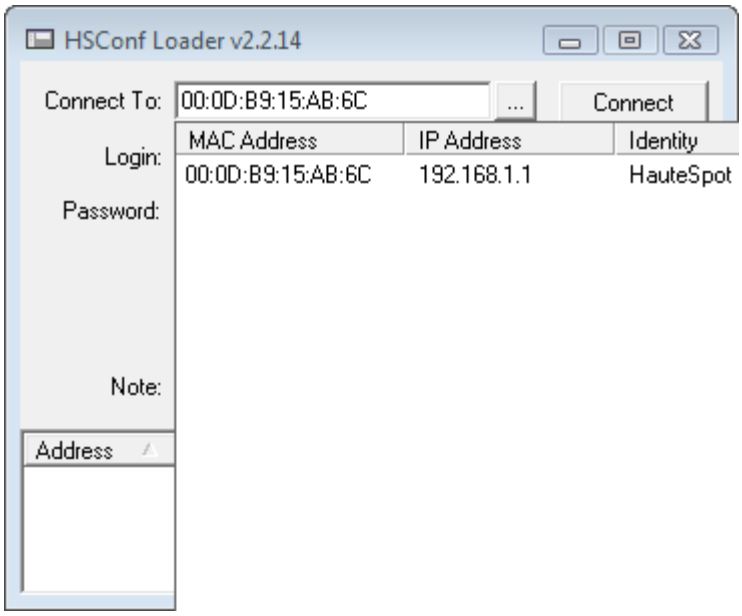
The **HSconf.exe** program opens the HSconf login window.



where:

-

  discovers and shows MNDP (HauteSpot Neighbor Discovery Protocol) or CDP (Cisco Discovery Protocol) devices.



- Connect

  logs on to the router by specified IP address (and the port number if you have changed it from the default value of 80) or MAC Address (if the router is in the same subnet), user name, and password.

- Save

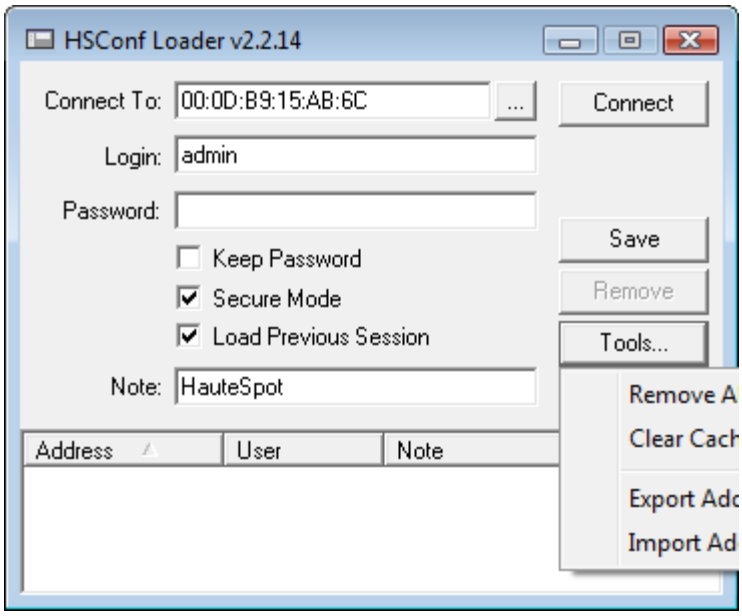
  saves the current sessions to the list (to run them, just double-click on an item).

- Remove

  removes selected item from the list.

- Tools...

  removes all items from the list, clears cache on the local disk, imports addresses from wbx file or exports them to wbx file.



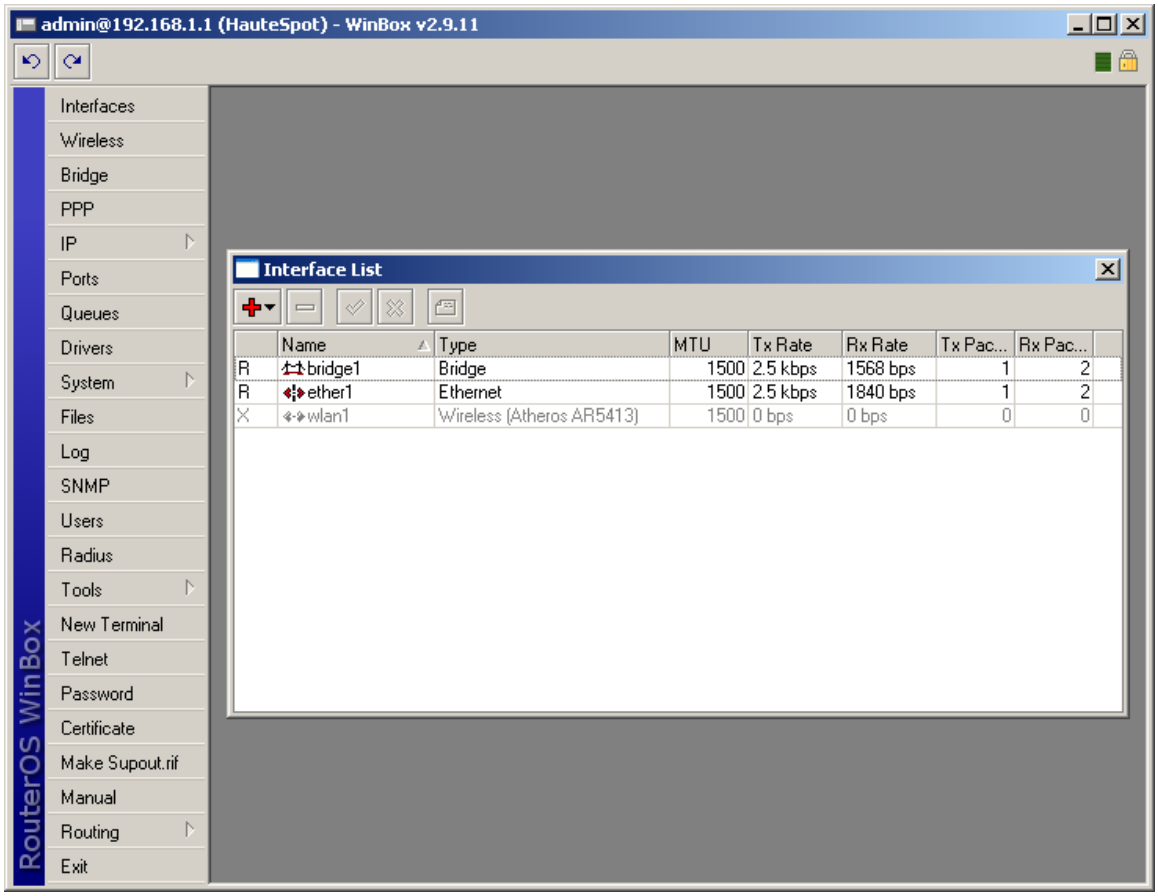
- Secure Mode

  provides privacy and data integrity between HSconf and RouterOS by means of TLS (Transport Layer Security) protocol.

- Keep Password

  Saves password as a plain text on a local hard drive. **Warning:** storing passwords in plain text allows anybody with access to your files to read the password from there.

The HSconf Console of the router:



The HSconf Console uses TCP port 8291. After logging onto the router you can work with the HauteSpot router's configuration through the HSconf console and perform the same tasks as using the regular console.

Overview of Common Functions

You can use the menu bar to navigate through the router's configuration menus, open configuration windows. By double clicking on some list items in the windows you can open configuration windows for the specific items, and so on.

There are some hints for using the HSconf Console:

- To open the required window, simply click on the corresponding menu item
-

  Add a new entry

-

  Remove an existing entry

-

  Enable an item

-

  Disable an item

-

  Make or edit a comment

-

  Refresh a window

-

  Undo an action

-

  Redo an action

-

  Logout from the HSconf Console

Troubleshooting

Description

- **Can I run HSconf on Linux?**
- Yes, you can run HSconf and connect to RouterOS, using Wine
- **I cannot open the HSconf Console**

  Check the port and address for **www** service in **/ip service print** list. Make sure the address you are connecting from matches the network you've specified in **address** field and that you've specified the correct port in the HSconf loader. The command **/ip service set www port=80 address=0.0.0.0/0** will change these values to the default ones so you will be able to connect specifying just the correct address of the router in the address field of HSconf loader

- The HSconf Console uses TCP port 8291. Make sure you have access to it through the firewall.

HauteRouterOS HTTP Proxy

Document revision: 1.1 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

The HauteRouterOS implements the following proxy server features:

- Regular HTTP proxy
- Transparent proxy. Can be transparent and regular at the same time
- Access list by source, destination, URL and requested method
- Direct Access List (specifies which resources should be accessed directly, and which - through another proxy server)
- Logging facility

Quick Setup Guide

To enable HTTP proxy, do the following:

```
[admin@Hautespot] ip proxy> set enabled=yes
[admin@Hautespot] ip proxy> print
                    enabled: yes
                       port: 8080
               parent-proxy: 0.0.0.0:0
  maximal-client-connecions: 1000
  maximal-server-connectons: 1000
[admin@Hautespot] ip proxy>
```

Remember to secure your proxy by preventing unauthorized access to it. Also you need to setup destination NAT in order to utilize transparent proxying facility:

```
[admin@Hautespot] ip firewall nat> add chain=dstnat protocol=tcp dst-port=80
action=redirect to-ports=8080
[admin@Hautespot] ip firewall nat> print
Flags: X - disabled, I - invalid, D - dynamic
 0   chain=dstnat protocol=tcp dst-port=80 action=redirect to-ports=8080
[admin@Hautespot] ip firewall nat>
```

Specifications

Packages required: ***system***
License required: *Level3*
Submenu level: ***/ip proxy***
Standards and Technologies: HTTP/1.0, HTTP/1.1, FTP

Description

This service performs proxying of HTTP and HTTP-proxy (for FTP, HTTP and HTTPS protocols) requests. It does not support caching transactions yet.

When setting up proxy service, make sure it serves only your clients, and is not misused as relay. Please read the security notice in the Access List Section!

Note that the proxy may be used as something like HTTP and FTP firewall (for example, denying access to mp3 files) or to redirect requests to external proxy (possibly, to a proxy with caching functions) transparently.

Setup

Submenu level: ***/ip proxy***

Property Description

enabled (yes | no; default: **no**) - whether the proxy server is enabled
maximal-client-connecions (*integer*; default: **1000**) - maximal number of connections accepted from clients (any further connections will be rejected)
maximal-server-connectons (*integer*; default: **1000**) - maximal number of connections made to servers (any further connections from clients will be put on hold until some server connections will terminate)
parent-proxy (*IP address*:*port*; default: **0.0.0.0:0**) - IP address and port of another HTTP proxy to redirect all requests to (exceptions may be defined in the "direct access" list
0.0.0.0:0 - no parent proxy is used
port (*port*; default: **8080**) - TCP port the proxy server will be listening on. This is to be specified on all clients that want to use the server as HTTP proxy. Transparent (with zero configuration for clients) proxy setup can be made by redirecting HTTP requests to this port in IP firewall using destination NAT feature

Notes

The web proxy listens to all IP addresses that the router has in its IP address list.

Example

To enable the proxy on port 8000:

```
[admin@Hautespot] ip proxy> set enabled=yes port=8000
[admin@Hautespot] ip proxy> print
                    enabled: yes
                       port: 8000
               parent-proxy: 0.0.0.0:0
  maximal-client-connecions: 1000
  maximal-server-connectons: 1000
[admin@Hautespot] ip proxy>
```

Access List

Submenu level: ***/ip proxy access***

Description

Access list is implemented in the same way as Hautespot RouterOS firewall rules. Rules are processed from the top to the bottom. First matching rule specifies decision of what to do with this connection. There is a total of 6 classifiers that specify matching constraints. If none of these classifiers is specified, the particular rule will match every connection.

If connection is matched by a rule, **action** property of this rule specifies whether connection will be allowed or not. If the particular connection does not match any rule, it will be allowed.

Property Description

action (allow | deny; default: **allow**) - specifies whether to pass or deny matched packets
dst-address (*IP address/netmask*) - destination address of the IP packet
dst-host (*wildcard*) - IP address or DNS name used to make connection the target server (this is the string user wrote in his/her browser before specifying port and path to a particular web page)
dst-port (*port*{1,10}) - a list or range of ports the packet is destined to
method (any | connect | delete | get | head | options | post | put | trace) - HTTP method used in the request (see HTTP Methods section in the end of this document)
path (*wildcard*) - name of the requested page within the target server (i.e. the name of a particular web page or document without the name of the server it resides on)
src-address (*IP address/netmask*) - source address of the IP packet

Notes

Wildcard properties (**dst-host** and **dst-path**) match a complete string (i.e., they will not match "example.com" if they are set to "example"). Available wildcards are '*' (match any number of any characters) and '?' (match any one character). Regular expressions are also accepted here, but if the property should be treated as a regular expression, it should start with a colon (':').

Small hits in using regular expressions:

- **** symbol sequence is used to enter **** character in console
- **\.** pattern means **.** only (in regular expressions single dot in pattern means any symbol)
- to show that no symbols are allowed before the given pattern, we use **^** symbol at the beginning of the pattern
- to specify that no symbols are allowed after the given pattern, we use **$** symbol at the end of the pattern
- to enter **[** or **]** symbols, you should escape them with backslash ****.

It is strongly recommended to deny all IP addresses except those behind the router as the proxy still may be used to access your internal-use-only (intranet) web servers. Also, consult examples in Firewall Manual on how to protect your router.

Direct Access List

Submenu level: ***/ip proxy direct***

Description

If **parent-proxy** property is specified, it is possible to tell proxy server whether to try to pass the request to the parent proxy or to resolve it connecting to the requested server directly. Direct Access List is managed just like Proxy Access List described in the previous chapter except the **action** argument.

Property Description

action (allow | deny; default: **allow**) - specifies the action to perform on matched packets
allow - always resolve matched requests directly bypassing the parent router
deny - resolve matched requests through the parent proxy. If no one is specified this has the same effect as **allow**
dst-address (*IP address/netmask*) - destination address of the IP packet
dst-host (*text*) - IP address or DNS name used to make connection the target server (this is the string user wrote in his/her browser before specifying port and path to a particular web page)
dst-port (*port*{1,10}) - a list or range of ports the packet is destined to
method (any | connect | delete | get | head | options | post | put | trace) - HTTP method used in the request (see HTTP Methods section in the end of this document)
path (*wildcard*) - name of the requested page within the target server (i.e. the name of a particular wed page or document without the name of the server it resides on)
src-address (*IP address/netmask*) - source address of the IP packet

Notes

Unlike the access list, the direct proxy access list has default action equal to **deny**. It takes place when no rules are specified or a particular request did not match any rule.

HTTP Methods

Description

OPTIONS

This method is a request of information about the communication options available on the chain between the client and the server identified by the **Request-URI**. The method allows the client to determine the options and (or) the requirements associated with a resource without initiating any resource retrieval

GET

This method retrieves whatever information identified by the **Request-URI**. If the **Request-URI** refers to a data processing process than the response to the **GET** method should contain data produced by the process, not the source code of the process procedure(-s), unless the source is the result of the process.

The **GET** method can become a *conditional* **GET** if the request message includes an **If-Modified-Since, If-Unmodified-Since, If-Match, If-None-Match**, or **If-Range** header field. The conditional **GET** method is used to reduce the network traffic specifying that the transfer of the entity should occur only under circumstances described by conditional header field(-s).

The **GET** method can become a *partial* **GET** if the request message includes a **Range** header field. The partial **GET** method intends to reduce unnecessary network usage by requesting only parts of entities without transferring data already held by client.

The response to a **GET** request is cacheable if and only if it meets the requirements for HTTP caching.

HEAD

This method shares all features of **GET** method except that the server must not return a message-body in the response. This retrieves the metainformation of the entity implied by the request which leads to a wide usage of it for testing hypertext links for validity, accessibility, and recent modification.

The response to a **HEAD** request may be cacheable in the way that the information contained in the response may be used to update previously cached entity identified by that **Request-URI**.

POST

This method requests that the origin server accept the entity enclosed in the request as a new subordinate of the resource identified by the **Request-URI**.

The actual action performed by the **POST** method is determined by the origin server and usually is **Request-URI** dependent.

Responses to **POST** method are not cacheable, unless the response includes appropriate **Cache-Control** or **Expires** header fields.

PUT

This method requests that the enclosed entity be stored under the supplied **Request-URI**. If another entity exists under specified **Request-URI**, the enclosed entity should be considered as updated (newer) version of that residing on the origin server. If the **Request-URI** is not pointing to an existing resource, the origin server should create a resource with that URI.

If the request passes through a cache and the **Request-URI** identifies one or more currently cached entities, those entries should be treated as stale. Responses to this method are not cacheable.

TRACE

This method invokes a remote, application-layer loop-back of the request message. The final recipient of the request should reflect the message received back to the client as the entity-body of a 200 (OK) response. The final recipient is either the origin server or the first proxy or gateway to receive a **Max-Forwards** value of **0** in the request. A **TRACE** request must not include an entity.

Responses to this method MUST NOT be cached.

Manual:Interface/HWMPplus

Applies to HauteRouterOS: 3, v4

- *Prerequisites for this article: you understand what WDS is and why to use it*
- *Software versions*: **3.28+** *(earlier versions are incompatible)*

Overview

HWMP+ is a HauteSpot specific layer-2 routing protocol for wireless mesh networks. It is based on Hybrid Wireless Mesh Protocol (HWMP) from IEEE 802.11s draft standard. It can be used instead of (Rapid) Spanning Tree protocols in mesh setups to ensure loop-free optimal routing.

The HWMP+ protocol however is not compatible with HWMP from IEEE 802.11s draft standard.

Note that the distribution system you use for your network need not to be Wireless Distribution System (WDS). HWMP+ mesh routing supports not only WDS interfaces, but also Ethernet interfaces inside the mesh. So you can use simple Ethernet based distribution system, or you can combine both WDS and Ethernet links!

Configuration

/interface mesh

Configure mesh interface.

admin-mac (MAC address, default: 00:00:00:00:00:00) -- administratively assigned MAC address, used when **auto-mac** setting is disabled

arp (disabled | enabled | proxy-arp | reply-only; default: enabled) - address resolution protocol setting

auto-mac (boolean, default: no) -- if disabled, then value from **admin-mac** will be used as the MAC address of the mesh interface; else address of some port will be used if ports are present

hwmp-default-hoplimit (integer: 1..255) -- maximum hop count for generated routing protocol packets; after a HWMP+ packet is forwarded "hoplimit" times, it is dropped

hwmp-prep-lifetime (time, default: 5m) -- lifetime for routes created from received PREP or PREQ messages

hwmp-preq-destination-only (boolean, default: yes) -- whether only destination can respond to HWMP+ PREQ message

hwmp-preq-reply-and-forward (boolean, default: yes) -- whether intermediate nodes should forward HWMP+ PREQ message after responding to it. Useful only when **hwmp-preq-destination-only** is disabled

hwmp-preq-retries (integer, default: 2) -- how much times to retry route discovery to a specific MAC address before the address is considered unreachable

hwmp-preq-waiting-time (time, default: 4s) -- how long to wait for a response to the first PREQ message. Note that for subsequent PREQs the waiting time is increased exponentially

hwmp-rann-interval (time, default: 10s) -- how often to send out HWMP+ RANN messages

hwmp-rann-lifetime (time, default: 1s) -- lifetime for routes created from received RANN messages

hwmp-rann-propagation-delay (number, default: 0.5) -- how long to wait before propagating a RANN message. Value in seconds

mesh-portal (boolean, default: no) -- whether this interface is a portal in the mesh network

mtu (number, default: 1500) -- maximum transmit units

name (string) -- interface name

reoptimize-paths (boolean, default: no) -- whether to send out periodic PREQ messages asking for known MAC addresses. Turing on this setting is useful if network topology is changing often. Note that if no reply is received to a reoptimization PREQ, the existing path is kept anyway (until it timeouts itself)

/interface mesh port

Configure mesh interface ports.

hello-interval (time, default: 10s) -- maximum interval between sending out HWMP+ Hello messages. Used only for Ethernet type ports

interface (interface name) -- interface name, which is to be included in a mesh

mesh (interface name) -- mesh interface this port belongs to

path-cost (integer: 0..65535; default: 10) -- path cost to the interface, used by routing protocol to determine the 'best' path

port-type (WDS | auto | ethernet | wireless) -- port type to use

- **auto** - port type is determined automatically based on the underlying interface's type
- **WDS** - a Wireless Distribution System interface, kind of point-to-point wireless link. Remote MAC address is known from wireless connection data
- **ethernet** - Remote MAC addresses are learned either from HWMP+ Hello messages or from source MAC addresses in received or forwarded traffic
- **wireless** - Remote MAC addresses are known from wireless connection data

active-port-type (read-only, wireless | WDS | ethernet-mesh | ethernet-bridge | ethernet-mixed) -- port type and state actually used

/interface mesh fdb

Read-only status of the mesh interface Forwarding Database (FDB).

mac-address (MAC address) -- MAC address corresponding for this FDB entry

seq-number (integer) -- sequence number used in routing protocol to avoid loops

type (local | outsider | direct | mesh | neighbor | larval | unknown) -- type of this FDB entry

- **local** -- MAC address belongs to the local router itself
- **outsider** -- MAC address belongs to a device external to the mesh network
- **direct** -- MAC address belongs to a wireless client on an interface that is in the mesh network
- **mesh** -- MAC address belongs to a device reachable over the mesh network; it can be either internal or external to the mesh network
- **neighbor** -- MAC address belongs to a mesh router that is direct neighbor to this router
- **larval** -- MAC address belongs to an unknown device that is reachable over the mesh network
- **unknown** -- MAC address belongs to an unknown device

mesh (interface name) -- the mesh interface this FDB entry belongs to

on-interface (interface name) -- mesh port used for traffic forwarding, kind of a next-hop value

lifetime (time) -- time remaining to live if this entry is not used for traffic forwarding

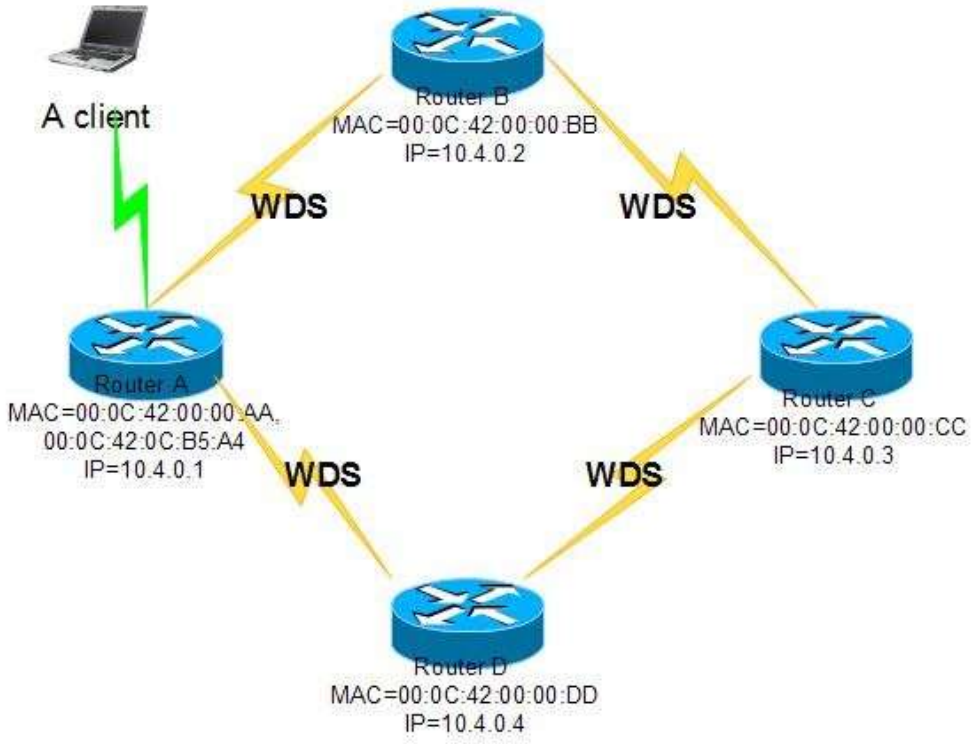
age (time) -- age of this FDB entry

metric (integer) -- metric value used by routing protocol to determine the 'best' path

Additional wireless configuration

Use **wds-default-cost** and **wds-cost-range** wireless interface parameters for controlling the metric that is used in the routing protocol. The WDS cost will be used as **path-cost** for ports dynamically added to the mesh interface.

Example



This example uses static WDS links that are dynamically added as mesh ports when they become active. Two different frequencies are used: one for AP interconnections, and one for client connections to APs, so the AP must have at least two wireless interfaces. Of course, the same frequency for all connections also could be used, but that might not work as good because of potential interference issues.

Repeat this configuration on all APs:

```
/interface mesh add disabled=no

/interface mesh port add interface=wlan1 mesh=mesh1

/interface mesh port add interface=wlan2 mesh=mesh1

# interface used for AP interconnections
/interface wireless set wlan1 disabled=no ssid=mesh frequency=2437
band=2.4ghz-b/g mode=ap-bridge \
  wds-mode=static-mesh wds-default-bridge=mesh1

# interface used for client connections
/interface wireless set wlan2 disabled=no ssid=mesh-clients
frequency=5180 band=5ghz mode=ap-bridge
```

```
# a static WDS interface for each AP you want to connect to
/interface wireless wds add disabled=no master-interface=wlan1
name=<descriptive name of remote end> \
  wds-address=<MAC address of remote end>
```

Here WDS interface is added manually, because static WDS mode is used. If you are using **wds-mode**=dynamic-mesh, all WDS interfaces will be created automatically. The **frequency** and **band** parameters are specified here only to produce valid example configuration; mesh protocol operations is by no means limited to, or optimized for, these particular values.

Note: You may want to increase **disconnect-timeout** wireless interface option to make the protocol more stable.

In real world setups you also should take care of securing the wireless connections, using **/interface wireless security-profile**. For simplicity that configuration it's not shown here.

Results on router A (there is one client is connected to wlan2):

```
[admin@A] > /interface mesh pr
Flags: X - disabled, R - running
 0  R name="mesh1" mtu=1500 arp=enabled mac-address=00:0C:42:0C:B5:A4
auto-mac=yes
      admin-mac=00:00:00:00:00:00 mesh-portal=no hwmp-default-
hoplimit=32
      hwmp-preq-waiting-time=4s hwmp-preq-retries=2 hwmp-preq-
destination-only=yes
      hwmp-preq-reply-and-forward=yes hwmp-prep-lifetime=5m hwmp-rann-
interval=10s
      hwmp-rann-propagation-delay=1s hwmp-rann-lifetime=22s

[admin@A] > interface mesh port p detail
Flags: X - disabled, I - inactive, D - dynamic
 0    interface=wlan1 mesh=mesh1 path-cost=10 hello-interval=10s port-
type=auto port-type-used=wireless
 1    interface=wlan2 mesh=mesh1 path-cost=10 hello-interval=10s port-
type=auto port-type-used=wireless
 2  D interface=router_B mesh=mesh1 path-cost=105 hello-interval=10s
port-type=auto port-type-used=WDS
 3  D interface=router_D mesh=mesh1 path-cost=76 hello-interval=10s
port-type=auto port-type-used=WDS
```

The FDB (Forwarding Database) at the moment contains information only about local MAC addresses, non-mesh nodes reachable through local interface, and direct mesh neighbors:

```
[admin@A] /interface mesh> fdb print
Flags: A - active, R - root
   MESH        TYPE     MAC-ADDRESS       ON-INTERFACE       LIFETIME
AGE
A  mesh1       local    00:0C:42:00:00:AA
3m17s
A  mesh1       neighbor 00:0C:42:00:00:BB router_B
1m2s
A  mesh1       neighbor 00:0C:42:00:00:DD router_D
3m16s
A  mesh1       direct   00:0C:42:0C:7A:2B wlan2
2m56s
A  mesh1       local    00:0C:42:0C:B5:A4
2m56s

[admin@A] /interface mesh> fdb print detail
Flags: A - active, R - root
 A  mac-address=00:0C:42:00:00:AA type=local age=3m21s mesh=mesh1
metric=0
     seqnum=4294967196
 A  mac-address=00:0C:42:00:00:BB type=neighbor on-interface=router_B
age=1m6s
    mesh=mesh1 metric=132 seqnum=4294967196
 A  mac-address=00:0C:42:00:00:DD type=neighbor on-interface=router_D
age=3m20s
     mesh=mesh1 metric=79 seqnum=4294967196
 A  mac-address=00:0C:42:0C:7A:2B type=direct on-interface=wlan2 age=3m
mesh=mesh1
     metric=10 seqnum=0
 A  mac-address=00:0C:42:0C:B5:A4 type=local age=3m mesh=mesh1 metric=0
seqnum=0
```

Test that ping works:

```
[admin@A] > /ping 00:0C:42:00:00:CC
00:0C:42:00:00:CC 64 byte ping time=108 ms
00:0C:42:00:00:CC 64 byte ping time=51 ms
00:0C:42:00:00:CC 64 byte ping time=39 ms
00:0C:42:00:00:CC 64 byte ping time=43 ms
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 39/60.2/108 ms
```

Router A had to discover path to Router C first, hence the slightly larger time for the first ping. Now the FDB also contains an entry for 00:0C:42:00:00:CC, with type "mesh".

Also test that ARP resolving works and so does IP level ping:

```
[admin@A] > /ping 10.4.0.3
10.4.0.3 64 byte ping: ttl=64 time=163 ms
10.4.0.3 64 byte ping: ttl=64 time=46 ms
10.4.0.3 64 byte ping: ttl=64 time=48 ms
3 packets transmitted, 3 packets received, 0% packet loss
```

```
round-trip min/avg/max = 46/85.6/163 ms
```

Mesh traceroute

There is also mesh traceroute command, that can help you to determine which paths are used for routing.

For example, for this network:

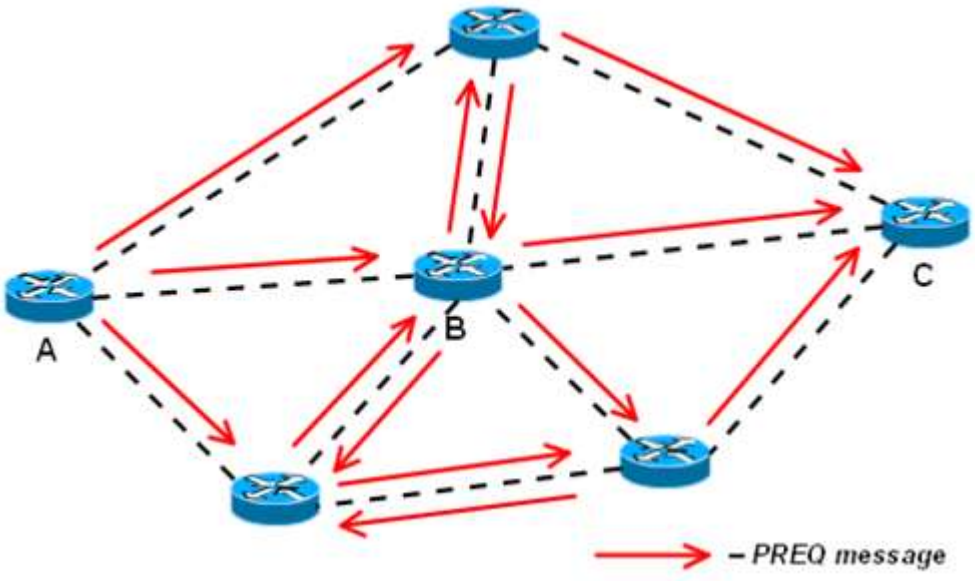
```
[admin@1] /interface mesh> fdb print
Flags: A - active, R - root
   MESH        TYPE     MAC-ADDRESS       ON-INTERFACE       LIFETIME
AGE
A  mesh1       local    00:0C:42:00:00:01
7m1s
A  mesh1       mesh     00:0C:42:00:00:02 wds4               17s
4s
A  mesh1       mesh     00:0C:42:00:00:12 wds4               4m58s
1s
A  mesh1       mesh     00:0C:42:00:00:13 wds4               19s
2s
A  mesh1       neighbor 00:0C:42:00:00:16 wds4
7m1s
A  mesh1       mesh     00:0C:42:00:00:24 wds4               18s
3s
```

Traceroute to 00:0C:42:00:00:12 shows:

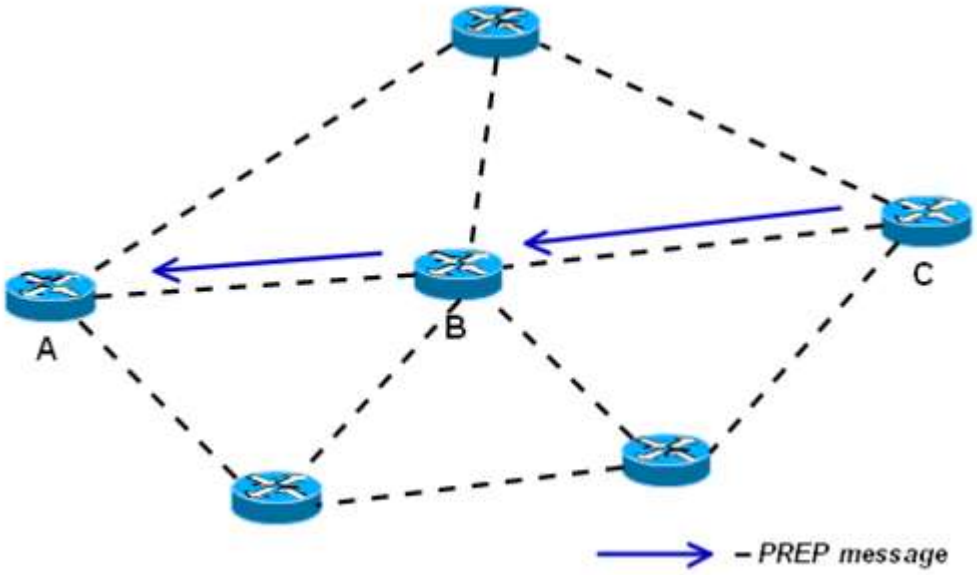
```
[admin@1] /interface mesh> traceroute mesh1 00:0C:42:00:00:12
ADDRESS           TIME         STATUS
00:0C:42:00:00:16 1ms          ttl-exceeded
00:0C:42:00:00:02 2ms          ttl-exceeded
00:0C:42:00:00:24 4ms          ttl-exceeded
00:0C:42:00:00:13 6ms          ttl-exceeded
00:0C:42:00:00:12 6ms          success
```

Protocol description

Reactive mode



Router A wants to discover path to C



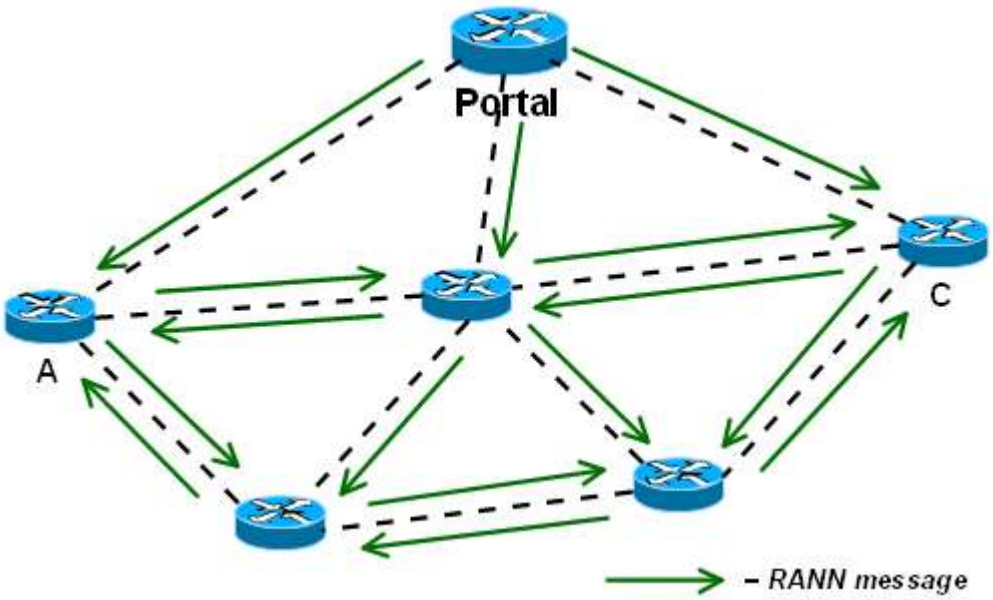
Router C sends unicast response to A

In reactive mode HWMP+ is very much like AODV (Ad-hoc On-demand Distance Vector). All path are discovered on demand, by flooding Path Request (PREQ) message in the network. The destination node or some router that has a path to the destionation will reply with a Path Response (PREP). Note that if the destination address belongs to a

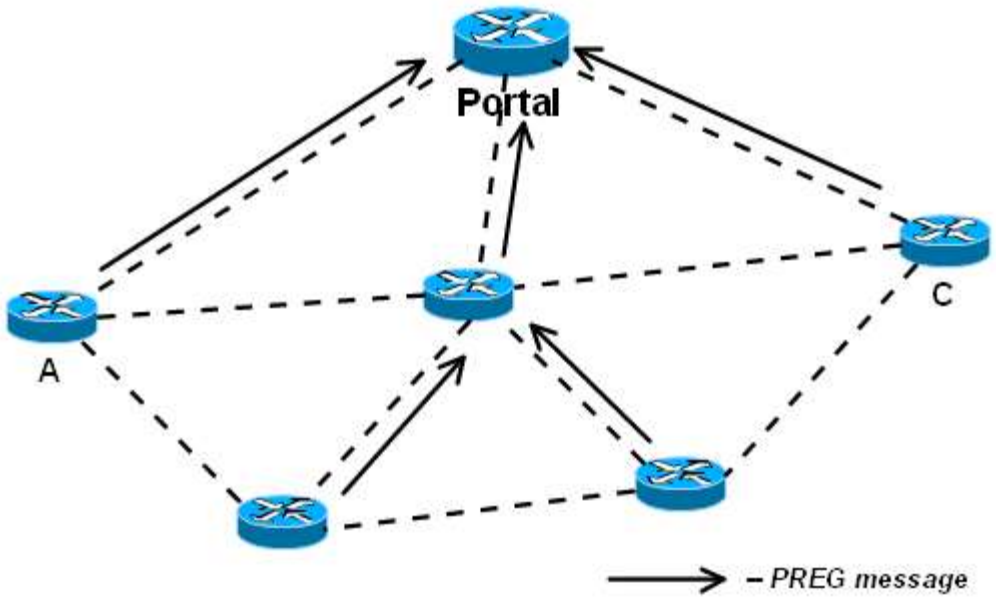
client, the AP this client is connected to will serve as proxy for him (i.e. reply to PREQs on his behalf).

This mode is best suited for mobile networks, and/or when most of the communication happens between intra-mesh nodes.

Proactive mode



The root announces itself by flooding RANN



Internal nodes respond with PREGs

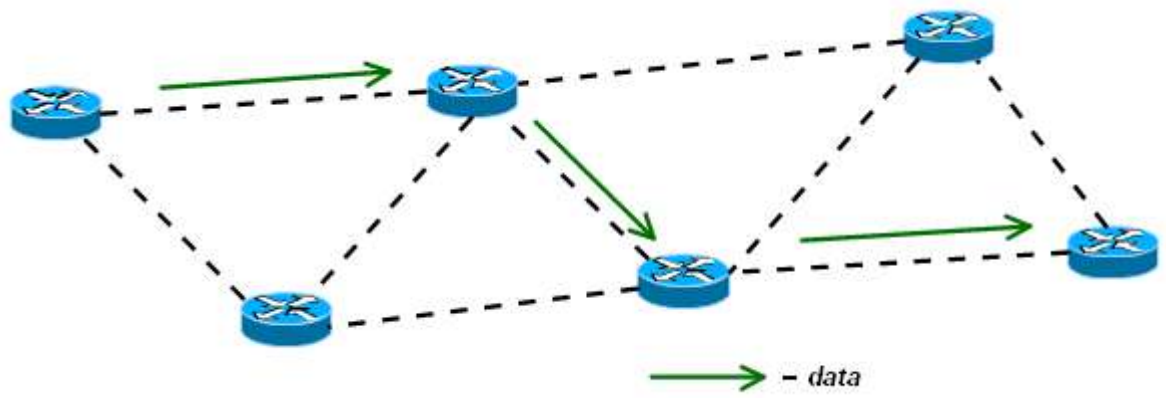
In proactive mode there are some routers configured as portals. In general being a portal means that router has interfaces to some other network,, i.e. it is entry/exit point to the mesh network.

The portals will announce their presence by flooding Root Announcement (RANN) message in the network. Internal nodes will reply with a Path Registration (PREG) message. The result of this process will be routing trees with roots in the portal.

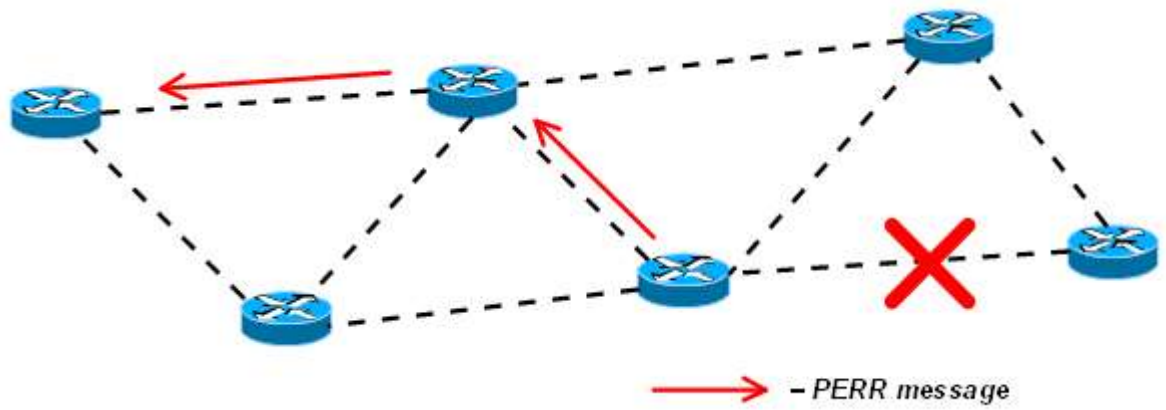
Routes to portals will serve as a kind of default routes. If an internal router does not know path to a particular destination, it will forward all data to its closest portal. The portal will then discover path on behalf of the router, if needed. The data afterwards will flow through the portal. This may lead to suboptimal routing, unless the data is addressed to the portal itself or some external network the portals has interfaces to.

Proactive mode is best suited when most of traffic goes between internal mesh nodes and a few portal nodes.

Topology change detection



Data flow path



After link disappears, error is propagated upstream

HWMP+ uses Path Error (PERR) message to notify that a link has disappeared. The message is propagated to all upstream nodes up to the data source. The source on PERR reception restarts path discovery process.

FAQ

Q. How is this better than RSTP?

A. It gives you optimal routing. RSTP is only for loop prevention.

Q. How the route selection is done?

A. The route with best metric is always selected after the discovery process. There is also a configuration option to periodically reoptimize already known routes.

Route metric is calculated as sum of individual link metrics.

Link metric is calculated in the same way as for (R)STP protocols:

- For Ethernet links the metric is configured statically (like for OSPF, for example).
- For WDS links the metric is updated dynamically depending on actual link bandwidth, which in turn is influenced by wireless signal strength, and the selected data transfer rate.

Currently the protocol does not take in account the amount of bandwidth being used on a link, but that might be also used in future.

Q. How is this better than OSPF/RIP/layer-3 routing in general?

A. WDS networks usually are bridged, not routed. The ability to self-configure is important for mesh networks; and routing generally requires much more configuration than bridging. Of you course, you always can run any L3 routing protocol over a bridged network, but for mesh networks that usually makes little sense.

Note: Since optimized layer-2 multicast forwarding is not included mesh protocol, it is better to avoid forwarding any multicast traffic (including OSPF) over meshed networks. If you need OSPF, then you have to configure OSPF NBMA neighbors that uses unicast instead.

Q. What about performance/CPU requirements?

A. The protocol itself, when properly configured, will take much less resources than OSPF (for example) would. Data forwarding performance on an individual router should be close to that of bridging.

Q. How does it work together with existing mesh setups that are using RSTP?

A. The internal structure of a RSTP network is transparent to the mesh protocol (because mesh hello packets are forwarded inside RSTP network). The mesh will see the path between two entry points in the RSTP network as a single segment. On the other hand, a mesh network is not transparent to the RSTP, since RSTP hello packets are not be forwarded inside the mesh network. *(This is the behaviour since 3.26)*

Warning: Routing loops are possible, if a mesh network is attached to a RSTP network in two or more points!

Note that if you have a WDS link between two access points, then both ends must have the same configuration (either as ports in a mesh on both ends, or as ports in a bridge interface on both ends).

You can also put a bridge interface as a mesh port (to be able to use bridge firewall, for example).

Q. Can I have multiple entry/exit points to the network?

A. If the entry/exit points are configured as portals (i.e. proactive mode is used), each router inside the mesh network will select its closest portal and forward all data to it. The portal will then discover path on behalf of the router, if needed.

Q. How to control or filter mesh traffic?

A. At the moment the only way is to use bridge firewall. Create a bridge interface, put the WDS interfaces and/or Ethernets in that bridge, and put that bridge in a mesh interface. Then configure bridge firewall rules.

To match MAC protocol used for mesh traffic encapsulation, use MAC protocol number 0x9AAA, and to mathc mesh routing tafffic, use MAC protocol number 0x9AAB. Example:

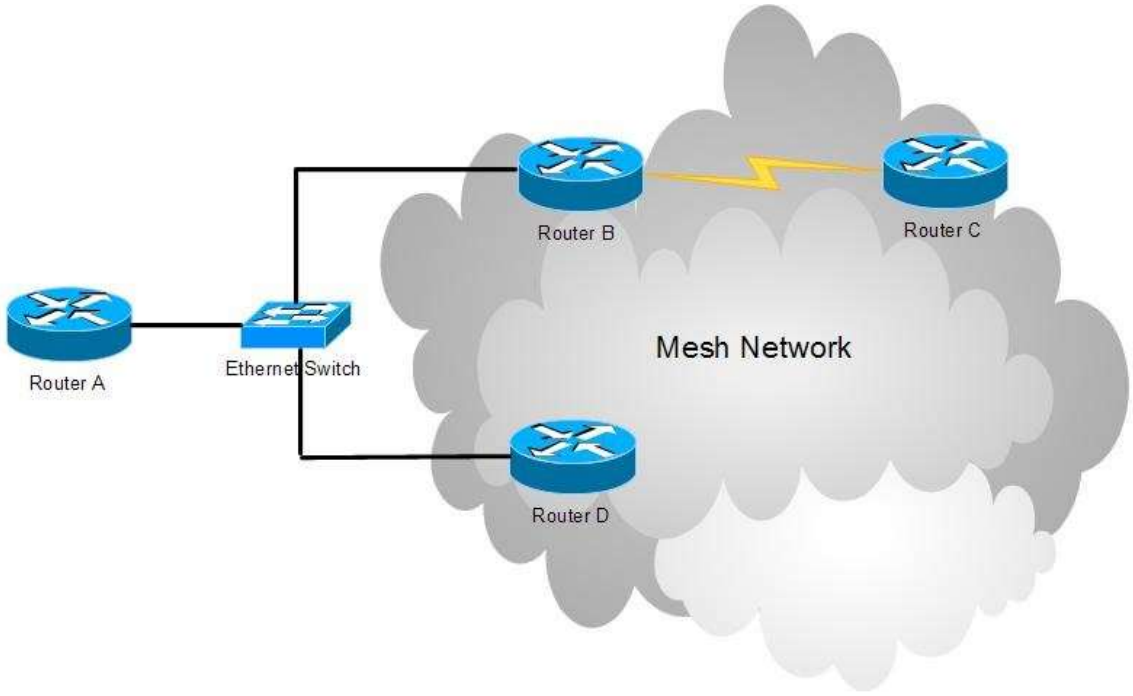
```
interface bridge settings set use-ip-firewall=yes
interface bridge filter add chain=input action=log mac-protocol=0x9aaa
interface bridge filter add chain=input action=log mac-protocol=0x9aab
```

Note that it is perfectly possible to create mixed mesh/bridge setups that will not work (e.g. *Problematic example 1* with bridge instead of switch). The recommended fail-safe way that will always work is to create a separate bridge interface per each physical interfaces; then add all these bridge interfaces as mesh ports.

Advanced topics

We all know that it's easy to make problematic layer-2 bridging or routing setups and hard to debug them. (Compared to layer-3 routing setups.) So there are a few bad configuration examples which could create problems for you. Avoid them!

Problematic example 1: Ethernet switch inside a mesh



Router A is outside the mesh, all the rest of routers: inside. For routers B, C, D all interfaces are added as mesh ports.

Router A will not be able to communicate reliably with router C. The problem manifests itself when D is the designated router for Ethernet; if B takes this role, everything is OK. The main cause of the problem is MAC address learning on Ethernet switch.

Consider what happens when router A wants to send something to C. We suppose router A either knowns or floods data to all interfaces. Either way, data arrives at switch. The switch, not knowing anything about destination's MAC address, forwards to data both to B and D.

What happens now:

1. B receives the packet on a mesh interface. Since the MAC address is not local for B, and B knows that he is not the designated router for the Ethernet network, he simply ignores the packet.
2. D receives the packet on a mesh interface. Since the MAC address is not local for B, and D is the designated router for the Ethernet network, he initiates path discovery process to C.

After path discovery is completed, D has information that C is reachable over B. Now D encapsulates the packet and forwards back to Ethernet network. The encapsulated packet forwarded by switch, received and forwarded by B, and received by C. So far everything is good.

Now C is likely to respond to the packet. Since B already knows where A is, he will decapsulate and forward the reply packet. But now switch will learn that the MAC address of C is reachable through B! That means, next time when something arrives from A addressed to C, the switch will forward data *only* to B. (And B, of course, will silently ignore the packet).

In contrast, if B took up the role of designated router, everything would be OK, because traffic would not have to go through the Ethernet switch twice.

Troubleshooting: either avoid such setup or disable MAC address learning on the switch. Note that on many switches it's not possible.

Also note that there will be no problem, if either:

- router A supports and is configured to use HWMP+;
- or Ethernet switch is replaced with and router that supports HWMP+ and has Ethernet interfaces added as mesh ports.

Problematic example 2: wireless modes

Consider this setup (invalid):

Routers A and B are inside the mesh, routers C: outside. For routers A and B all interfaces are added as mesh ports.

It is not possible to bridge wlan1 and wlan2 on router B now. The reason for this is pretty obvious if you understand how WDS works. For WDS communications four address frames are used. This is because for wireless multihop forwarding you need to know both the addresses of intermediate hops, as well as the original sender and final receiver. In contrast, non-WDS 802.11 communication includes only three MAC addresses in a frame. That's why it's not possible to do multihop forwarding in station mode.

Troubleshooting: depends on what you want to achieve:

1. If you want to router C as a repeater either for wireless or Ethernet traffic, configure WDS link between router B and router C, and run mesh routing protocol on all nodes.
2. In other cases configure wlan2 on router B in AP mode, and wlan on router C in station mode.

HauteRouterOS ICMP Bandwidth Test

Document revision: 1.2 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

The ICMP Bandwidth Tester (Ping Speed) can be used to approximately evaluate the throughput to **any** remote computer and thereby help to discover network 'bottlenecks'.

Specifications

Packages required: ***advanced-tools***
License required: *Level1*
Submenu level: ***/tool***
Standards and Technologies: ICMP (RFC792)
Hardware usage: *Not significant*

ICMP Bandwidth Test

Description

The ICMP test uses two standard echo-requests per second. The time between these pings can be changed. Ping packet size variation makes it possible to approximately evaluate connection parameters and speed with different packet sizes. Statistics for throughput is calculated using the size of the ICMP packet, the interval between ICMP echo-request and echo-reply and the differences between parameters of the first and the second packet.

Property Description

do (*name*) - assigned name of the script to start
first-ping-size (*integer*: 32..64000; default: **32**) - first ICMP packet size
second-ping-size (*integer*: 32..64000; default: **1500**) - second ICMP packet size
time-between-pings (*integer*) - the time between the first and the second ICMP echo-requests in seconds. A new ICMP-packet pair will never be sent before the previous pair is completely sent and the algorithm itself will never send more than two requests in one second

once - specifies that the ping will be performed only once
interval (*time*: 20ms..5s) - time interval between two ping repetitions

Example

In the following example we will test the bandwidth to a host with IP address **159.148.60.2**. The interval between repetitions will be **1** second.

```
[admin@Hautespot] tool> ping-speed 159.148.60.2 interval=1s
    current: 2.23Mbps
    average: 2.61Mbps

[admin@Hautespot] tool>
```

HauteRouterOS Interface Bonding

Document revision: 1.1 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

Bonding is a technology that allows to aggregate multiple ethernet-like interfaces into a single virtual link, thus getting higher data rates and providing failover.

Quick Setup Guide

Let us assume that we have 2 NICs in each router (**Router1** and **Router2**) and want to get maximum data rate between 2 routers. To make this possible, follow these steps:

1. Make sure that you do not have IP addresses on interfaces which will be enslaved for bonding interface!
2. Add **bonding** interface on **Router1**:

   ```
   [admin@Router1] interface bonding> add slaves=ether1,ether2
   ```

   And on **Router2**:

```
[admin@Router2] interface bonding> add slaves=ether1,ether2
```

3. Add addresses to bonding interfaces:

```
[admin@Router1] ip address> add address=172.16.0.1/24
interface=bonding1
[admin@Router2] ip address> add address=172.16.0.2/24
interface=bonding1
```

4. Test the link from **Router1**:

```
[admin@Router1] interface bonding> /pi 172.16.0.2
172.16.0.2 ping timeout
172.16.0.2 ping timeout
172.16.0.2 ping timeout
172.16.0.2 64 byte ping: ttl=64 time=2 ms
```

```
172.16.0.2 64 byte ping: ttl=64 time=2 ms
```

Note that bonding interface needs a couple of seconds to get connectivity with its peer.

Specifications

Packages required: ***system***
License required: *Level1*
Submenu level: ***/interface bonding***
Standards and Technologies: None
Hardware usage: *Not significant*

Related Documents

- Linux Ethernet Bonding Driver mini-howto

Description

To provide a proper failover, you should specify **link-monitoring** parameter. It can be:

- MII (Media Independent Interface) type1 or type2 - Media Independent Interface is an abstract layer between the operating system and the NIC which detects whether the link is running (it performs also other functions, but in our case this is the most important).
- ARP - Address Resolution Protocol periodically (for **arp-interval** time) checks the link status.

link-monitoring is used to check whether the link is up or not.

Property Description

arp (disabled | enabled | proxy-arp | reply-only; default: **enabled**) - Address Resolution Protocol for the interface
disabled - the interface will not use ARP
enabled - the interface will use ARP
proxy-arp - the interface will use the ARP proxy feature
reply-only - the interface will only reply to the requests originated to its own IP addresses. Neighbour MAC addresses will be resolved using **/ip arp** statically set table only
arp-interval (*time*; default: **00:00:00.100**) - time in milliseconds which defines how often to monitor ARP requests
arp-ip-targets (*IP address*; default: **""**) - IP target address which will be monitored if **link-monitoring** is set to **arp**. You can specify multiple IP addresses, separated by comma
down-delay (*time*; default: **00:00:00**) - if a link failure has been detected, bonding interface is disabled for **down-delay** time. Value should be a multiple of **mii-interval**
lacp-rate (1sec | 30secs; default: **30secs**) - Link Aggregation Control Protocol rate specifies how often to exchange with LACPDUs between bonding peer. Used to determine whether link is up or other changes have occured in the network. LACP tries to adapt to these changes providing failover.
link-monitoring (arp | mii-type1 | mii-type2 | none; default: **none**) - method to use for monitoring the link (whether it is up or down)
arp - uses Address Resolution Protocol to determine whether the remote interface is reachable
mii-type1 - uses Media Independent Interface type1 to determine link status. Link status determenation relies on the device driver. If bonding shows that the link status is up, when it should not be, then it means that this card don't support this possibility.
mii-type2 - uses MII type2 to determine link status (used if **mii-type1** is not supported by the NIC)
none - no method for link monitoring is used. If a link fails, it is not considered as down (but no traffic passes through it, thus).
mac-address (*read-only: MAC address*) - MAC address of the bonding interface
mii-interval (*time*; default: **00:00:00.100**) - how often to monitor the link for failures (parameter used only if **link-monitoring** is **mii-type1** or **mii-type2**)
mtu (*integer*: 68..1500; default: **1500**) - Maximum Transmit Unit in bytes
mode (802.3ad | active-backup | balance-alb | balance-rr | balance-tlb | balance-xor | broadcast; default: **balance-rr**) - interface bonding mode. Can be one of:
802.3ad - IEEE 802.3ad dynamic link aggregation. In this mode, the interfaces are aggregated in a group where each slave shares the same speed. If you use a switch between 2 bonding routers, be sure that this switch supports IEEE 802.3ad standard. Provides fault tolerance and load balancing.
active-backup - provides link backup. Only one slave can be active at a time. Another slave becomes active only, if first one fails.
balance-alb - adaptive load balancing. It includes **balance-tlb** and received traffic is also balanced. Device driver should support for setting the mac address, then it is active. Otherwise **balance-alb** doesn't work. No special switch is required.
balance-rr - round-robin load balancing. Slaves in bonding interface will transmit and receive data in sequential order. Provides load balancing and fault tolerance.
balance-tlb - Outgoing traffic is distributed according to the current load on each slave. Incoming traffic is received by the current slave. If receiving slave fails, then another slave takes the MAC address of the failed slave. Doesn't require any special switch support.
balance-xor - Use XOR policy for transmit. Provides only failover (in very good quality), but not load balancing, yet.

broadcast - Broadcasts the same data on all interfaces at once. This provides fault tolerance but slows down traffic throughput on some slow machines.
name (*name*) - descriptive name of bonding interface
primary (*name*; default: **none**) - Interface is used as primary output media. If primary interface fails, only then others slaves will be used. This value works only with **mode=active-backup**
slaves (*name*) - at least two ethernet-like interfaces separated by a comma, which will be used for bonding
up-delay (*time*; default: **00:00:00**) - if a link has been brought up, bonding interface is disabled for **up-delay** time and after this time it is enabled. Value should be a multiple of **mii-interval**

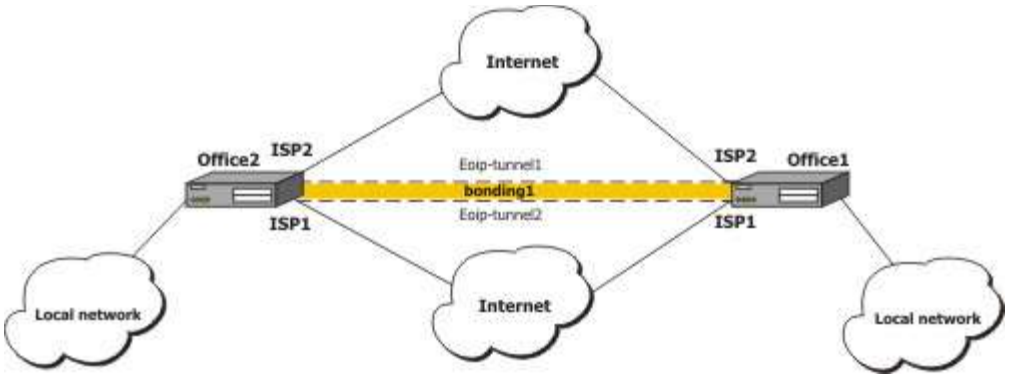
Notes

Link failure detection and failover is working significantly better with expensive network cards, for example, made by Intel, then with more cheap ones. For example, on Intel cards failover is taking place in less than a second after link loss, while on some other cards, it may require up to 20 seconds. Also, the Active load balancing (mode=**balance-alb**) does not work on some cheap cards.

Application Examples

Bonding two Eoip tunnels

Assume you need to configure the HauteSpot router for the following network setup, where you have two offices with 2 ISP for each. You want combine links for getting double speed and provide failover:



We are assuming that connections to Internet through two ISP are configured for both routers.

- Configuration on routers
  - on **Office1**

```
[admin@office1] > /interface print
```

```
Flags: X - disabled, D - dynamic, R - running
 #    NAME                          TYPE             RX-RATE
TX-RATE    MTU
 0  R isp1                          ether            0          0       1500
 1  R isp2                          ether            0          0       1500

[admin@office1] > /ip address print
Flags: X - disabled, I - invalid, D - dynamic
 #   ADDRESS            NETWORK         BROADCAST       INTERFACE
 0   1.1.1.1/24         1.1.1.0         1.1.1.255       isp2
 1   10.1.0.111/24      10.1.0.0        10.1.0.255      isp1
```

- on **Office2**

```

[admin@office2] interface> print
Flags: X - disabled, D - dynamic, R - running
 #    NAME                          TYPE             RX-RATE
TX-RATE   MTU
 0  R isp2                          ether            0          0
1500
 1  R isp1                          ether            0          0
1500
[admin@office2] interface> /ip add print
Flags: X - disabled, I - invalid, D - dynamic
 #   ADDRESS            NETWORK         BROADCAST       INTERFACE
 0   2.2.2.1/24         2.2.2.0         2.2.2.255       isp2
 1   10.1.0.112/24      10.1.0.0        10.1.0.255      isp1
```

- Eoip tunnel confguration
  - for **Office1** through ISP1

```
[admin@office1] > interface eoip add remote-address=10.1.0.112
tunnel-id=2
\... mac-address=FE:FD:00:00:00:04
[admin@office1] > interface eoip print
Flags: X - disabled, R - running
0  R name="eoip-tunnel2" mtu=1500 mac-address==FE:FD:00:00:00:04
arp=enabled
\... remote-address=10.1.0.112 tunnel-id=2
```

  - for **Office2** through ISP1

```

```

```
[admin@office2] > interface eoip add remote-address=10.1.0.111
tunnel-id=2
\... mac-address=FE:FD:00:00:00:02
[admin@office2] > interface eoip print
Flags: X - disabled, R - running
0  R name="eoip-tunnel2" mtu=1500 mac-address=FE:FD:00:00:00:02
arp=enabled
\... remote-address=10.1.0.111 tunnel-id=2
```

- for **Office1**through ISP2

```
[admin@office1] > interface eoip add remote-address=2.2.2.1
tunnel-id=1
\... mac-address=FE:FD:00:00:00:03
[admin@office1] interface eoip> print
Flags: X - disabled, R - running
 0  R name="eoip-tunnel1" mtu=1500 mac-address=FE:FD:00:00:00:03
arp=enabled
      remote-address=2.2.2.1 tunnel-id=1

 1  R name="eoip-tunnel2" mtu=1500 mac-address=FE:FD:00:00:00:04
arp=enabled
      remote-address=10.1.0.112 tunnel-id=2
```

- for **Office2**through ISP2

```

[admin@office2] > interface eoip add remote-address=1.1.1.1
tunnel-id=1
\... mac-address=FE:FD:00:00:00:01
[admin@office2] interface eoip> print
Flags: X - disabled, R - running
 0  R name="eoip-tunnel1" mtu=1500 mac-address=FE:FD:00:00:00:01
arp=enabled
      remote-address=1.1.1.1 tunnel-id=1

 1  R name="eoip-tunnel2" mtu=1500 mac-address=FE:FD:00:00:00:02
arp=enabled
      remote-address=10.1.0.111 tunnel-id=2
```

- Bonding confguration
  - for **Office1**

```

```

```
[admin@office1] interface bonding> add slaves=eoip-tunnel1,eoip-
tunnel2
[admin@office1] interface bonding> print
Flags: X - disabled, R - running
 0  R name="bonding1" mtu=1500 mac-address=00:0C:42:03:20:E7
arp=enabled
       slaves=eoip-tunnel1,eoip-tunnel2 mode=balance-rr
primary=none
       link-monitoring=none arp-interval=00:00:00.100 arp-ip-
targets=""
       mii-interval=00:00:00.100 down-delay=00:00:00 up-
delay=00:00:00
       lacp-rate=30secs
[admin@office1] ip address> add address=3.3.3.1/24
interface=bonding1
[admin@office1] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
 #   ADDRESS            NETWORK         BROADCAST       INTERFACE
 0   1.1.1.1/24         1.1.1.0         1.1.1.255       isp2
 1   10.1.0.111/24      10.1.0.0        10.1.0.255      isp1
 2   3.3.3.1/24         3.3.3.0         3.3.3.255       bonding1

```

- for **Office2**

```

[admin@office2] interface bonding> add slaves=eoip-tunnel1,eoip-
tunnel2
[admin@office2] interface bonding> print
Flags: X - disabled, R - running
 0  R name="bonding1" mtu=1500 mac-address=00:0C:42:03:20:E7
arp=enabled
       slaves=eoip-tunnel1,eoip-tunnel2 mode=balance-rr
primary=none
       link-monitoring=none arp-interval=00:00:00.100 arp-ip-
targets=""
       mii-interval=00:00:00.100 down-delay=00:00:00 up-
delay=00:00:00
       lacp-rate=30secs
[admin@office2] ip address> add address=3.3.3.2/24
interface=bonding1
[admin@office2] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
 #   ADDRESS            NETWORK         BROADCAST       INTERFACE
 0   2.2.2.1/24         2.2.2.0         2.2.2.255       isp2
 1   10.1.0.112/24      10.1.0.0        10.1.0.255      isp1
 2   3.3.3.2/24         3.3.3.0         3.3.3.255       bonding1
[admin@office2] ip address> /ping 3.3.3.1
3.3.3.1 64 byte ping: ttl=64 time=2 ms
3.3.3.1 64 byte ping: ttl=64 time=2 ms
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 2/2.0/2 ms
```

HauteRouterOS IP accounting

Document revision: 2.1 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

Authentication, Authorization and Accounting feature provides a possibility of local and/or remote (on RADIUS server) Point-to-Point and HotSpot user management and traffic accounting (all IP traffic passing the router is accounted; local traffic accounting is an option).

Specifications

Packages required: ***system***
License required: *Level1*
Submenu level: ***/user***, ***/ppp***, ***/ip accounting***, ***/radius***
Standards and Technologies: RADIUS
Hardware usage: *Traffic accounting requires additional memory*

Local IP Traffic Accounting

Submenu level: ***/ip accounting***

Description

As each packet passes through the router, the packet source and destination addresses are matched against an IP pair in the accounting table and the traffic for that pair is increased. The traffic of PPP, PPTP, PPPoE, ISDN and HotSpot clients can be accounted on per-user basis too. Both the number of packets and the number of bytes are accounted.

If no matching IP or user pair exists, a new entry will be added to the table

Only the packets that enter and leave the router are accounted. Packets that are dropped in the router are not counted. Packets that are NATted on the router will be accounted for with the actual IP addresses on each side. Packets that are going

through bridged interfaces (i.e. inside the bridge interface) are also accounted correctly.

Traffic, generated by the router itself, and sent to it, may as well be accounted.

Property Description

enabled (yes | no; default: **no**) - whether local IP traffic accounting is enabled
account-local-traffic (yes | no; default: **no**) - whether to account the traffic to/from the router itself
threshold (*integer*; default: **256**) - maximum number of IP pairs in the accounting table (maximal value is **8192**)

Notes

For bidirectional connections two entries will be created.

Each IP pair uses approximately 100 bytes

When the threshold limit is reached, no new IP pairs will be added to the accounting table. Each packet that is not accounted in the accounting table will then be added to the **uncounted** counter!

Example

Enable IP accounting:

```
[admin@Hautespot] ip accounting> set enabled=yes
[admin@Hautespot] ip accounting> print
                enabled: yes
  account-local-traffic: no
              threshold: 256
[admin@Hautespot] ip accounting>
```

Example

See the uncounted packets:

```
[admin@Hautespot] ip accounting uncounted> print
    packets: 0
      bytes: 0
[admin@Hautespot] ip accounting uncounted>
```

Local IP Traffic Accounting Table

Submenu level: ***/ip accounting snapshot***

Description

When a snapshot is made for data collection, the accounting table is cleared and new IP pairs and traffic data are added. The more frequently traffic data is collected, the less likelihood that the IP pairs thereshold limit will be reached.

Property Description

bytes (*read-only: integer*) - total number of bytes, matched by this entry
dst-address (*read-only: IP address*) - destination IP address
dst-user (*read-only: text*) - recipient's name (if aplicable)
packets (*read-only: integer*) - total number of packets, matched by this entry
src-address (*read-only: IP address*) - source IP address
src-user (*read-only: text*) - sender's name (if aplicable)

Notes

Usernames are shown only if the users are connected to the router via a PPP tunnel or are authenticated by HotSpot.

Before the first snapshot is taken, the table is empty.

Example

To take a new snapshot:

```
[admin@Hautespot] ip accounting snapshot> take
[admin@Hautespot] ip accounting snapshot> print
 # SRC-ADDRESS     DST-ADDRESS     PACKETS    BYTES      SRC-USER    DST-USER
 0 192.168.0.2     159.148.172.197 474        19130
 1 192.168.0.2     10.0.0.4        3          120
 2 192.168.0.2     192.150.20.254  32         3142
 3 192.150.20.254  192.168.0.2     26         2857
 4 10.0.0.4        192.168.0.2     2          117
 5 159.148.147.196 192.168.0.2     2          136
 6 192.168.0.2     159.148.147.196 1          40
 7 159.148.172.197 192.168.0.2     835        1192962
[admin@Hautespot] ip accounting snapshot>
```

Web Access to the Local IP Traffic Accounting Table

Submenu level: ***/ip accounting web-access***

Description

The web page report make it possible to use the standard Unix/Linux tool wget to collect the traffic data and save it to a file or to use Hautespot shareware Traffic Counter to display the table. If the web report is enabled and the web page is viewed, the **snapshot** will be made when connection is initiated to the web page. The **snapshot** will be displayed on the web page. TCP protocol, used by http connections with the wget tool guarantees that none of the traffic data will be lost. The **snapshot** image will be made when the connection from wget is initiated. Web browsers or wget should connect to URL: **http://routerIP/accounting/ip.cgi**

Property Description

accessible-via-web (yes | no; default: **no**) - wheather the snapshot is available via web
address (*IP address/netmask*; default: **0.0.0.0**) - IP address range that is allowed to access the snapshot

Example

To enable web access from **10.0.0.1** server only:

```
[admin@Hautespot] ip accounting web-access> set accessible-via-web=yes \
\... address=10.0.0.1/32
[admin@Hautespot] ip accounting web-access> print
    accessible-via-web: yes
               address: 10.0.0.1/32
[admin@Hautespot] ip accounting web-access>
```

HauteRouterOS IP Pools

Document revision: 0.0 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

IP pools are used to define range of IP addresses that is used for DHCP server and Point-to-Point servers

Specifications

Packages required: ***system***
License required: *Level1*
Submenu level: ***/ip pool***
Standards and Technologies: none
Hardware usage: *Not significant*

Description

IP pools simply group IP addresses for further usage. It is a single configuration point for all features that assign IP addresses to clients.

Notes

Whenever possible, the same ip address is given out to each client (OWNER/INFO pair).

Setup

Submenu level: ***/ip pool***

Property Description

name (*name*) - the name of the pool
next-pool (*name*) - when address is acquired from pool that has no free addresses, and *next-pool* property is set to another pool, then next IP address will be acquired from *next-pool*
ranges (*IP address*) - IP address list of non-overlapping IP address ranges in form of: **from1-to1,from2-to2,...,fromN-toN**. For example, **10.0.0.1-10.0.0.27,10.0.0.32-10.0.0.47**

Example

To define a pool named **ip-pool** with the **10.0.0.1-10.0.0.125** address range excluding gateway's address **10.0.0.1** and server's address **10.0.0.100**, and the other pool **dhcp-pool**, with the **10.0.0.200-10.0.0.250** address range:

```
[admin@Hautespot] ip pool> add name=ip-pool ranges=10.0.0.2-
10.0.0.99,10.0.0.101
10.0.0.126
[admin@Hautespot] ip pool> add name=dhcp-pool ranges=10.0.0.200-10.0.0.250
[admin@Hautespot] ip pool> print
  # NAME                                      RANGES
  0 ip-pool                                   10.0.0.2-10.0.0.99
                                              10.0.0.101-10.0.0.126
  1 dhcp-pool                                 10.0.0.200-10.0.0.250

[admin@Hautespot] ip pool>
```

Used Addresses from Pool

Submenu level: ***/ip pool used***

Description

Here you can see all used IP addresses from IP pools.

Property Description

pool (*read-only: name*) - name of the IP pool
address (*read-only: IP address*) - IP address that is assigned to client form the pool
owner (*read-only: MAC address*) - MAC address of the client
info (*read-only: name*) - name of the interface to which the client is connected to

Example

See used addresses from pool:

```
[admin@Hautespot] ip pool used> print
POOL  ADDRESS         OWNER                     INFO
local 192.168.0.100   00:0C:42:03:1F:60         test
local 192.168.0.99    00:0C:42:03:21:0F         test
```

HauteRouterOS IP Security

Document revision: 3.4 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Specifications

Packages required: ***security***
License required: *Level1*
Submenu level: ***/ip ipsec***
Standards and Technologies: IPsec
Hardware usage: *consumes a lot of CPU time (Intel Pentium MMX or AMD K6 suggested as a minimal configuration)*

Description

IPsec (IP Security) supports secure (encrypted) communications over IP networks.

Encryption

After packet is src-natted, but before putting it into interface queue, IPsec policy database is consulted to find out if packet should be encrypted. Security Policy Database (SPD) is a list of rules that have two parts:

- **Packet matching** - packet source/destination, protocol and ports (for TCP and UDP) are compared to values in policy rules, one after another
- **Action** - if rule matches action specified in rule is performed:
  - **accept** - continue with packet as if there was no IPsec
  - **drop** - drop packet
  - **encrypt** - encrypt packet

Each SPD rule can be associated with several Security Associations (SA) that determine packet encryption parameters (key, algorithm, SPI).

Note that packet can only be encrypted if there is usable SA for policy rule. By setting SPD rule security "level" user can control what happens when there is no valid SA for policy rule:

- **use** - if there is no valid SA, send packet unencrypted (like accept rule)
- **acquire** - send packet unencrypted, but ask IKE daemon to establish new SA
- **require** - drop packet, and ask IKE daemon to establish new SA.

Decryption

When encrypted packet is received for local host (after **dst-nat** and **input** filter), the appropriate SA is looked up to decrypt it (using packet source, destination, security protocol and SPI value). If no SA is found, the packet is dropped. If SA is found, packet is decrypted. Then decrypted packet's fields are compared to policy rule that SA is linked to. If the packet does not match the policy rule it is dropped. If the packet is decrypted fine (or authenticated fine) it is "received once more" - it goes through **dst-nat** and routing (which finds out what to do - either forward or deliver locally) again.

Note that before **forward** and **input** firewall chains, a packet that was not decrypted on local host is compared with SPD reversing its matching rules. If SPD requires encryption (there is valid SA associated with matching SPD rule), the packet is dropped. This is called incoming policy check.

Internet Key Exchange

The Internet Key Exchange (IKE) is a protocol that provides authenticated keying material for Internet Security Association and Key Management Protocol (ISAKMP) framework. There are other key exchange schemes that work with ISAKMP, but IKE is the most widely used one. Together they provide means for authentication of hosts and automatic management of security associations (SA).

Most of the time IKE daemon is doing nothing. There are two possible situations when it is activated:

- There is some traffic caught by a policy rule which needs to become encrypted or authenticated, but the policy doesn't have any SAs. The policy notifies IKE daemon about that, and IKE daemon initiates connection to remote host.
- IKE daemon responds to remote connection.

In both cases, peers establish connection and execute 2 phases:

- **Phase 1** - The peers agree upon algorithms they will use in the following IKE messages and authenticate. The keying material used to derive keys for all SAs and to protect following ISAKMP exchanges between hosts is generated also.
- **Phase 2** - The peers establish one or more SAs that will be used by IPsec to encrypt data. All SAs established by IKE daemon will have lifetime values (either limiting time, after which SA will become invalid, or amount of data that can be encrypted by this SA, or both).

There are two lifetime values - soft and hard. When SA reaches it's soft lifetime treshold, the IKE daemon receives a notice and starts another phase 2 exchange to replace this SA with fresh one. If SA reaches hard lifetime, it is discarded.

IKE can optionally provide a Perfect Forward Secrecy (PFS), whish is a property of key exchanges, that, in turn, means for IKE that compromising the long term phase 1 key will not allow to easily gain access to all IPsec data that is protected by SAs established through this phase 1. It means an additional keying material is generated for each phase 2.

Generation of keying material is computationally very expensive. *Exempli gratia*, the use of modp8192 group can take several seconds even on very fast computer. It usually takes place once per phase 1 exchange, which happens only once between any host pair and then is kept for long time. PFS adds this expensive operation also to each phase 2 exchange.

Diffie-Hellman MODP Groups

Diffie-Hellman (DH) key exchange protocol allows two parties without any initial shared secret to create one securely. The following Modular Exponential (MODP) Diffie-Hellman (also known as "Oakley") Groups are supported:

Diffie-Hellman Group	Modulus	Reference
Group 1	768 bits	RFC2409
Group 2	1024 bits	RFC2409
Group 5	1536 bits	RFC3526

IKE Traffic

To avoid problems with IKE packets hit some SPD rule and require to encrypt it with not yet established SA (that this packet perhaps is trying to establish), locally originated packets with UDP source port 500 are not processed with SPD. The same way packets with UDP destination port 500 that are to be delivered locally are not processed in incoming policy check.

Setup Procedure

To get IPsec to work with automatic keying using IKE-ISAKMP you will have to configure **policy**, **peer** and **proposal** (optional) entries.

For manual keying you will have to configure **policy** and **manual-sa** entries.

Policy Settings

Submenu level: ***/ip ipsec policy***

Description

Policy table is needed to determine whether encryption should be applied to a packet.

Property Description

action (accept | drop | encrypt; default: **accept**) - specifies what action to undertake with a packet that matches the policy
accept - pass the packet
drop - drop the packet
encrypt - apply transformations specified in this policy and it's SA
decrypted (*integer*) - how many incoming packets were decrypted by the policy
dont-fragment (clear | inherit | set; default: **clear**) - The state of the **don't fragment** IP header field
clear - clear (unset) the fields, so that packets previously marked as **don't fragment** got fragmented
inherit - do not change the field
set - set the field, so that each packet matching the rule will not be fragmented
dst-address (*IP address*/*netmask*:*port*; default: **0.0.0.0/32:any**) - destination IP address
encrypted (*integer*) - how many outgoing packets were encrypted by the policy
in-accepted (*integer*) - how many incoming packets were passed through by the policy without an attempt to decrypt
in-dropped (*integer*) - how many incoming packets were dropped by the policy without an attempt to decrypt
ipsec-protocols (*multiple choice:* ah | esp; default: **esp**) - specifies what combination of Authentication Header and Encapsulating Security Payload protocols you want to apply to matched traffic. AH is applied after ESP, and in case of tunnel mode ESP will be applied in tunnel mode and AH - in transport mode

level (acquire | require | use; default: **require**) - specifies what to do if some of the SAs for this policy cannot be found:
use - skip this transform, do not drop packet and do not acquire SA from IKE daemon
acquire - skip this transform, but acquire SA for it from IKE daemon
require - drop packet but acquire SA
manual-sa (*name*; default: **none**) - name of manual-sa template that will be used to create SAs for this policy
none - no manual keys are set
not-decrypted (*integer*) - how many incoming packets the policy attempted to decrypt. but discarded for any reason
not-encrypted (*integer*) - how many outgoing packets the policy attempted to encrypt. but discarded for any reason
out-accepted (*integer*) - how many outgoing packets were passed through by the policy without an attempt to encrypt
out-dropped (*integer*) - how many outgoing packets were dropped by the policy without an attempt to encrypt
ph2-state (*read-only:* expired | no-phase2 | established) - indication of the progress of key establishing
expired - there are some leftovers from previous phase2. In general it is similar to **no-phase2**
no-phase2 - no keys are estabilished at the moment
estabilished - Appropriate SAs are in place and everything should be working fine
proposal (*name*; default: **default**) - name of proposal information that will be sent by IKE daemon to establish SAs for this policy
protocol (*name* | *integer*; default: **all**) - protocol name or number
sa-dst-address (*IP address*; default: **0.0.0.0**) - SA destination IP address
sa-src-address (*IP address*; default: **0.0.0.0**) - SA source IP address
src-address (*IP address*/*netmask*:*port*; default: **0.0.0.0/32:any**) - source IP address
tunnel (yes | no; default: **no**) - specifies whether to use tunnel mode

Notes

All packets are IPIP encapsulated in tunnel mode, and their new IP header **src-address** and **dst-address** are set to **sa-src-address** and **sa-dst-address** values of this policy. If you do not use tunnel mode (*id est* you use transport mode), then only packets whose source and destination addresses are the same as **sa-src-address** and **sa-dst-address** can be processed by this policy. Transport mode can only work with packets that originate at and are destined for IPsec peers (hosts that established security associations). To encrypt traffic between networks (or a network and a host) you have to use tunnel mode.

It is good to have **dont-fragment** cleared because encrypted packets are always bigger than original and thus they may need fragmentation.

If you are using IKE to establish SAs automatically, then policies on both routers must exactly match each other, *id est* **src-address=1.2.3.0/27** on one router and **dst-address=1.2.3.0/28** on another would not work. Source address values on one router MUST be equal to destination address values on the other one, and vice versa.

Example

To add a policy to encrypt all the traffic between two hosts (10.0.0.147 and 10.0.0.148), we need do the following:

```
[admin@WiFi] ip ipsec policy> add sa-src-address=10.0.0.147 \
\... sa-dst-address=10.0.0.148 action=encrypt
[admin@WiFi] ip ipsec policy> print
Flags: X - disabled, D - dynamic, I - invalid
 0   src-address=10.0.0.147/32:any dst-address=10.0.0.148/32:any protocol=all
     action=encrypt level=require ipsec-protocols=esp tunnel=no
     sa-src-address=10.0.0.147 sa-dst-address=10.0.0.148 proposal=default
     manual-sa=none dont-fragment=clear

[admin@WiFi] ip ipsec policy>
```

to view the policy statistics, do the following:

```
[admin@WiFi] ip ipsec policy> print stats
Flags: X - disabled, D - dynamic, I - invalid
  0   src-address=10.0.0.147/32:any dst-address=10.0.0.148/32:any
      protocol=all ph2-state=no-phase2 in-accepted=0 in-dropped=0
      out-accepted=0 out-dropped=0 encrypted=0 not-encrypted=0 decrypted=0
      not-decrypted=0

[admin@WiFi] ip ipsec policy>
```

Peers

Submenu level: ***/ip ipsec peer***

Description

Peer configuration settings are used to establish connections between IKE daemons (phase 1 configuration). This connection then will be used to negotiate keys and algorithms for SAs.

Property Description

address (*IP address*/*netmask*:*port*; default: **0.0.0.0/32:500**) - address prefix. If remote peer's address matches this prefix, then this peer configuration is used while authenticating and establishing phase 1. If several peer's addresses matches several configuration entries, the most specific one (i.e. the one with largest netmask) will be used
dh-group (*multiple choice:* modp768 | modp1024 | modp1536; default: **esp**) - Diffie-Hellman MODP group (cipher strength)

enc-algorithm (*multiple choice:* des | 3des | aes-128 | aes-192 | aes-256; default: **3des**) - encryption algorithm. Algorithms are named in strength increasing order
exchange-mode (*multiple choice:* main | aggressive | base; default: **main**) - different ISAKMP phase 1 exchange modes according to RFC 2408.DO not use other modes then **main** unless you know what you are doing
generate-policy (yes | no; default: **no**) - allow this peer to establish SA for non-existing policies. Such policies are created dynamically for the lifetime of SA. This way it is possible, for example, to create IPsec secured L2TP tunnels, or any other setup where remote peer's IP address is not known at configuration time
hash-algorithm (*multiple choice:* md5 | sha; default: **md5**) - hashing algorithm. SHA (Secure Hash Algorithm) is stronger, but slower
lifebytes (*integer*; default: **0**) - phase 1 lifetime: specifies how much bytes can be transferred before SA is discarded
0 - SA expiration will not be due to byte count excess
lifetime (*time*; default: **1d**) - phase 1 lifetime: specifies how long the SA will be valid; SA will be discarded after this time
proposal-check (*multiple choice:* claim | exact | obey | strict; default: **strict**) - phase 2 lifetime check logic:
claim - take shortest of proposed and configured lifetimes and notify initiator about it
exact - require lifetimes to be the same
obey - accept whatever is sent by an initiator
strict - If proposed lifetime IS longer than default then reject proposal otherwise accept proposed lifetime
secret (*text*; default: **""**) - secret string. If it starts with '0x', it is parsed as a hexadecimal value
send-initial-contact (yes | no; default: **yes**) - specifies whether to send initial IKE information or wait for remote side

Notes

AES (Advanced Encryption Standard) encryption algorithms are much faster than DES, so it is recommended to use this algorithm class whenever possible. But, AES's speed is also its drawback as it potentially can be cracked faster, so use AES-256 when you need security or AES-128 when speed is also important.

Both peers MUST have the same encryption and authentication algorithms, DH group and exchange mode. Some legacy hardware may support only DES and MD5.

You should set **generate-policy** flag to **yes** only for trusted peers, because there is no verification done for the established policy. To protect yourself against possible unwanted events, add policies with **action=accept** for all networks you don't want to be encrypted at the top of policy list. Since dynamic policies are added at the bottom of the list, they will not be able to override your configuration.

Example

To define new peer configuration for **10.0.0.147** peer with **secret=gwejimezyfopmekun**:

```
[admin@WiFi] ip ipsec peer>add address=10.0.0.147/32 \
\... secret=gwejimezyfopmekun
[admin@WiFi] ip ipsec peer> print
Flags: X - disabled
  0   address=10.0.0.147/32:500 secret="gwejimezyfopmekun" generate-policy=no
      exchange-mode=main send-initial-contact=yes proposal-check=obey
      hash-algorithm=md5 enc-algorithm=3des dh-group=modp1024 lifetime=1d
      lifebytes=0

[admin@WiFi] ip ipsec peer>
```

Remote Peer Statistics

Submenu level: ***/ip ipsec remote-peers***

Description

This submenu provides you with various statistics about remote peers that currently have established phase 1 connections with this router. Note that if peer doesn't show up here, it doesn't mean that no IPsec traffic is being exchanged with it. For example, manually configured SAs will not show up here.

Property Description

estabilished (*read-only: text*) - shows date and time when phase 1 was established with the peer
local-address (*read-only: IP address*) - local ISAKMP SA address
ph2-active (*read-only: integer*) - how many phase 2 negotiations with this peer are currently taking place
ph2-total (*read-only: integer*) - how many phase 2 negotiations with this peer took place
remote-address (*read-only: IP address*) - peer's IP address
side (*multiple choice, read-only:* initiator | responder) - shows which side initiated the connection
initiator - phase 1 negotiation was started by this router
responder - phase 1 negotiation was started by peer
state (*read-only: text*) - state of phase 1 negotiation with the peer
estabilished - normal working state

Example

To see currently estabilished SAs:

```
[admin@WiFi] ip ipsec> remote-peers print
  0 local-address=10.0.0.148 remote-address=10.0.0.147 state=established
    side=initiator established=jan/25/2003 03:34:45 ph2-active=0 ph2-total=1
[admin@WiFi] ip ipsec>
```

Installed SAs

Submenu level: ***/ip ipsec installed-sa***

Description

This facility provides information about installed security associations including the keys

Property Description

add-lifetime (*read-only: time*) - soft/hard expiration time counted from installation of SA
auth-algorithm (*multiple choice, read-only:* none | md5 | sha1) - authentication algorithm used in SA
auth-key (*read-only: text*) - authentication key presented in form of hex string
current-addtime (*read-only: text*) - time when this SA was installed
current-bytes (*read-only: integer*) - amount of data processed by this SA's crypto algorithms
current-usetime (*read-only: text*) - time when this SA was first used
direction (*multiple choice, read-only:* in | out) - SA direction
dst-address (*read-only: IP address*) - destination address of SA taken from respective policy
enc-algorithm (*multiple choice, read-only:* none | des | 3des | aes) - encryption algorithm used in SA
enc-key (*read-only: text*) - encryption key presented in form of hex string (not applicable to AH SAs)
lifebytes (*read-only: integer*) - soft/hard expiration threshold for amount of processed data
replay (*read-only: integer*) - size of replay window presented in bytes. This window protects the receiver against replay attacks by rejecting old or duplicate packets.
spi (*read-only: integer*) - SPI value of SA, represented in hexadecimal form
src-address (*read-only: IP address*) - source address of SA taken from respective policy
state (*multiple choice, read-only:* larval | mature | dying | dead) - SA living phase
use-lifetime (*read-only: time*) - soft/hard expiration time counted from the first use of SA

Example

Sample printout looks as follows:

```
[admin@WiFi] ip ipsec> installed-sa print
Flags: A - AH, E - ESP, P - pfs, M - manual
  0 E   spi=E727605 direction=in src-address=10.0.0.148
        dst-address=10.0.0.147 auth-algorithm=sha1 enc-algorithm=3des
        replay=4 state=mature
        auth-key="ecc5f4aee1b297739ec88e324d7cfb8594aa6c35"
```

```
        enc-key="d6943b8ea582582e449bde085c9471ab0b209783c9eb4bbd"
        add-lifetime=24m/30m use-lifetime=0s/0s lifebytes=0/0
        current-addtime=jan/28/2003 20:55:12
        current-usetime=jan/28/2003 20:55:23 current-bytes=128
  1 E   spi=E15CEE06 direction=out src-address=10.0.0.147
        dst-address=10.0.0.148 auth-algorithm=sha1 enc-algorithm=3des
        replay=4 state=mature
        auth-key="8ac9dc7ecebfed9cd1030ae3b07b32e8e5cb98af"
        enc-key="8a8073a7afd0f74518c10438a0023e64cc660ed69845ca3c"
        add-lifetime=24m/30m use-lifetime=0s/0s lifebytes=0/0
        current-addtime=jan/28/2003 20:55:12
        current-usetime=jan/28/2003 20:55:12 current-bytes=512
[admin@WiFi] ip ipsec>
```

Flushing Installed SA Table

Command name: ***/ip ipsec installed-sa flush***

Description

Sometimes after incorrect/incomplete negotiations took place, it is required to flush manually the installed SA table so that SA could be renegotiated. This option is provided by the **flush** command.

Property Description

sa-type (*multiple choice:* ah | all | esp; default: **all**) - specifies SA types to flush
ah - delete AH protocol SAs only
esp - delete ESP protocol SAs only
all - delete both ESP and AH protocols SAs

Example

To flush all the SAs installed:

```
[admin@Hautespot] ip ipsec installed-sa> flush
[admin@Hautespot] ip ipsec installed-sa> print
[admin@Hautespot] ip ipsec installed-sa>
```

Counters

Submenu level: ***/ip ipsec counters***

Property Description

in-accept (*read-only: integer*) - shows how many incoming packets were matched by **accept** policy
in-accept-isakmp (*read-only: integer*) - shows how many incoming UDP packets on port 500 were let through without matching a policy
in-decrypted (*read-only: integer*) - shows how many incoming packets were successfully decrypted
in-drop (*read-only: integer*) - shows how many incoming packets were matched by **drop** policy (or **encrypt** policy with **level=require** that does not have all necessary SAs)
in-drop-encrypted-expected (*read-only: integer*) - shows how many incoming packets were matched by **encrypt** policy and dropped because they were not encrypted
out-accept (*read-only: integer*) - shows how many outgoing packets were matched by **accept** policy (including the default "accept all" case)
out-accept-isakmp (*read-only: integer*) - shows how many locally originated UDP packets on source port 500 (which is how ISAKMP packets look) were let through without policy matching
out-drop (*read-only: integer*) - shows how many outgoing packets were matched by **drop** policy (or **encrypt** policy with **level=require** that does not have all necessary SAs)
out-encrypt (*read-only: integer*) - shows how many outgoing packets were encrypted successfully

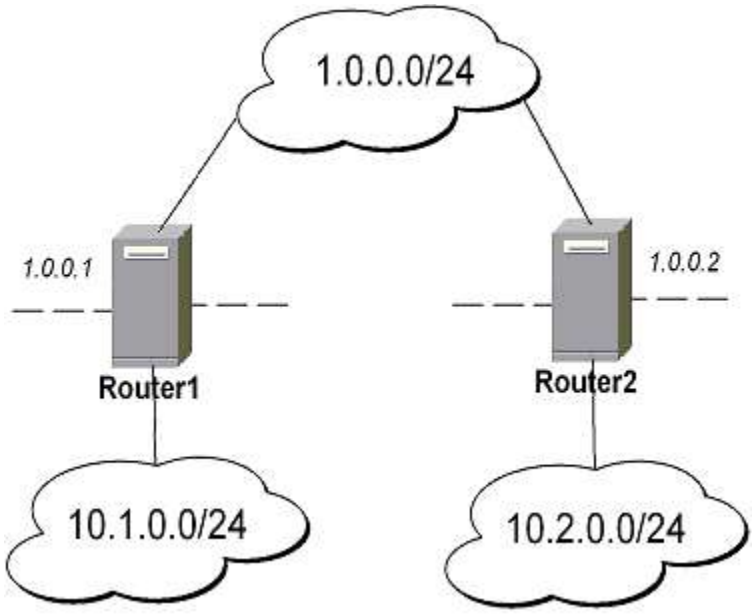
Example

To view current statistics:

```
[admin@WiFi] ip ipsec> counters print
                    out-accept: 6
               out-accept-isakmp: 0
                      out-drop: 0
                   out-encrypt: 7
                     in-accept: 12
                in-accept-isakmp: 0
                         in-drop: 0
                    in-decrypted: 7
    in-drop-encrypted-expected: 0
[admin@WiFi] ip ipsec>
```

Application Examples

Hautespot Router to Hautespot Router



- transport mode example using ESP with automatic keying
  - for **Router1**

```
[admin@Router1] > ip ipsec policy add sa-src-address=1.0.0.1 sa-dst-address=1.0.0.2 \
\... action=encrypt
[admin@Router1] > ip ipsec peer add address=1.0.0.2 \
```

```
\... secret="gvejimezyfopmekun"
```

  - for **Router2**

```
[admin@Router2] > ip ipsec policy add sa-src-address=1.0.0.2 sa-dst-address=1.0.0.1 \
\... action=encrypt
[admin@Router2] > ip ipsec peer add address=1.0.0.1 \
```

```
\... secret="gvejimezyfopmekun"
```

- transport mode example using ESP with automatic keying and automatic policy generating on Router 1 and static policy on Router 2
  - for **Router1**

```
[admin@Router1] > ip ipsec peer add address=1.0.0.0/24 \
```

```
\... secret="gvejimezyfopmekun" generate-policy=yes
```

  - for **Router2**

```
[admin@Router2] > ip ipsec policy add sa-src-address=1.0.0.2 sa-
dst-address=1.0.0.1 \
\... action=encrypt
[admin@Router2] > ip ipsec peer add address=1.0.0.1 \
\... secret="gvejimezyfopmekun"
```

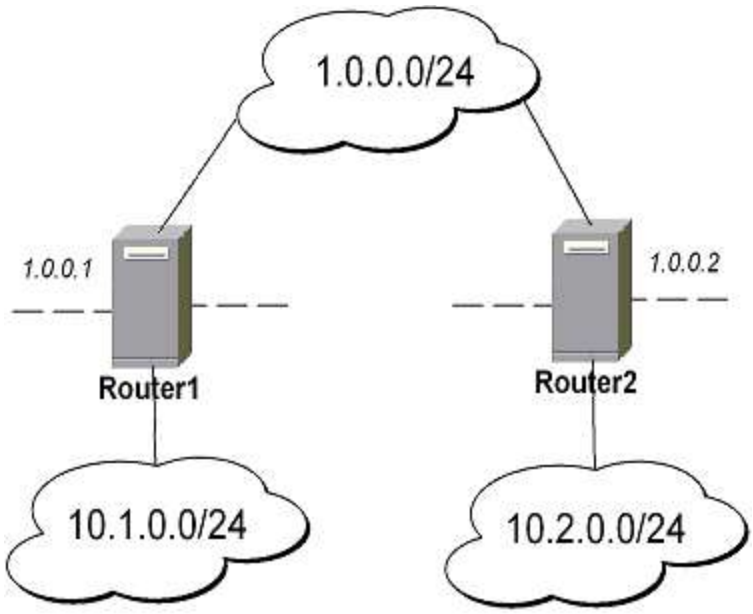
- tunnel mode example using AH with manual keying
  - for **Router1**

```
[admin@Router1] > ip ipsec manual-sa add name=ah-sa1 \
\... ah-spi=0x101/0x100 ah-key=abcfed
[admin@Router1] > ip ipsec policy add src-address=10.1.0.0/24 \
\... dst-address=10.2.0.0/24 action=encrypt ipsec-protocols=ah \
\... tunnel=yes sa-src=1.0.0.1 sa-dst=1.0.0.2 manual-sa=ah-sa1
```

  - for **Router2**

```
[admin@Router2] > ip ipsec manual-sa add name=ah-sa1 \
\... ah-spi=0x100/0x101 ah-key=abcfed
[admin@Router2] > ip ipsec policy add src-address=10.2.0.0/24 \
\... dst-address=10.1.0.0/24 action=encrypt ipsec-protocols=ah \
\... tunnel=yes sa-src=1.0.0.2 sa-dst=1.0.0.1 manual-sa=ah-sa1
```

IPsec Between two Masquerading Hautespot Routers



1. Add accept and masquerading rules in SRC-NAT
    - for **Router1**

```
[admin@Router1] > ip firewall nat \
\... add src-address=10.1.0.0/24 dst-address=10.2.0.0/24
[admin@Router1] > ip firewall nat add out-interface=public \
```

```
\... action=masquerade
```

    - for **Router2**

```
[admin@Router2] > ip firewall nat \
\... add src-address=10.2.0.0/24 dst-address=10.1.0.0/24
[admin@Router2] > ip firewall nat add out-interface=public \
```

```
\... action=masquerade
```

2. configure IPsec
    - for **Router1**

```
[admin@Router1] > ip ipsec policy add src-address=10.1.0.0/24 \
\... dst-address=10.2.0.0/24 action=encrypt tunnel=yes \
\... sa-src-address=1.0.0.1 sa-dst-address=1.0.0.2
[admin@Router1] > ip ipsec peer add address=1.0.0.2 \
```

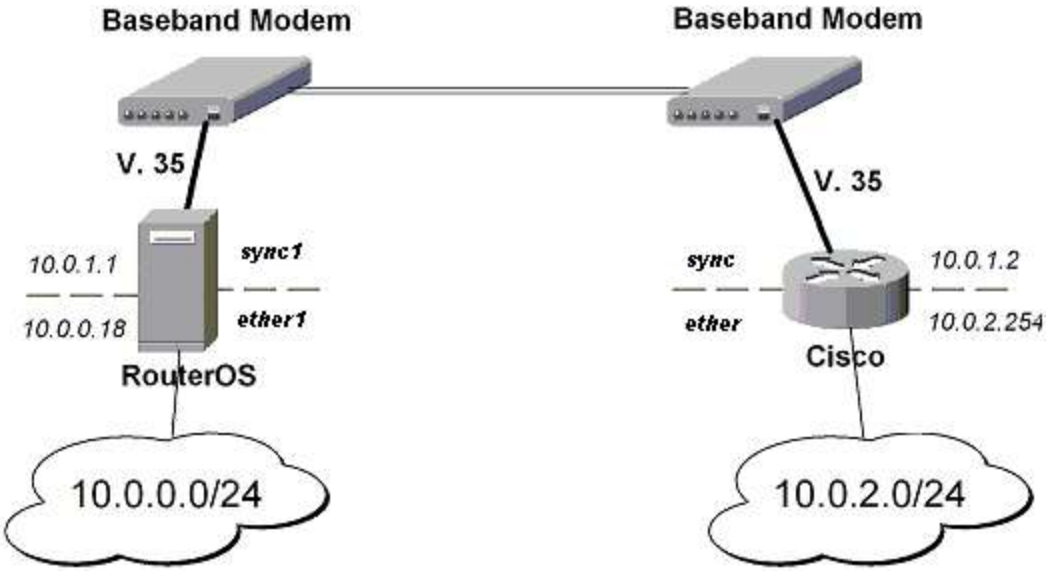
```
\... exchange-mode=aggressive secret="gvejimezyfopmekun"
```

    - for **Router2**

```
[admin@Router2] > ip ipsec policy add src-address=10.2.0.0/24 \
\... dst-address=10.1.0.0/24 action=encrypt tunnel=yes \
\... sa-src-address=1.0.0.2 sa-dst-address=1.0.0.1
[admin@Router2] > ip ipsec peer add address=1.0.0.1 \
```

```
\... exchange-mode=aggressive secret="gvejimezyfopmekun"
```

Hautespot router to CISCO Router



We will configure IPsec in tunnel mode in order to protect traffic between attached subnets.

1. Add peer (with phase1 configuration parameters), DES and SHA1 will be used to protect IKE traffic
   - for **Hautespot** router

```
[admin@Hautespot] > ip ipsec peer add address=10.0.1.2 \
```

```
\... secret="gvejimezyfopmekun" enc-algorithm=des
```

   - for **CISCO** router

```
! Configure ISAKMP policy (phase1 config, must match
configuration
! of "/ip ipsec peer" on RouterOS). Note that DES is default
! encryption algorithm on Cisco. SHA1 is default authentication
! algorithm
crypto isakmp policy 9
  encryption des
  authentication pre-share
  group 2
  hash md5
  exit

! Add preshared key to be used when talking to RouterOS
```

```
crypto isakmp key gvejimezyfopmekun address 10.0.1.1
255.255.255.255
```

2. Set encryption proposal (phase2 proposal - settings that will be used to encrypt actual data) to use DES to encrypt data

- for **Hautespot** router

```
[admin@Hautespot] > ip ipsec proposal set default enc-
algorithms=des
```

- for **CISCO** router

```
! Create IPsec transform set - transformations that should be
applied to
! traffic - ESP encryption with DES and ESP authentication with
SHA1
! This must match "/ip ipsec proposal"
crypto ipsec transform-set myset esp-des esp-sha-hmac
  mode tunnel
```

```
  exit
```

3. Add policy rule that matches traffic between subnets and requires encryption with ESP in tunnel mode
   - for **Hautespot** router

```
[admin@Hautespot] > ip ipsec policy add \
\... src-address=10.0.0.0/24 dst-address=10.0.2.0/24
action=encrypt \
```

```
\... tunnel=yes sa-src=10.0.1.1 sa-dst=10.0.1.2
```

   - for **CISCO** router

```
! Create access list that matches traffic that should be
encrypted
access-list 101 permit ip 10.0.2.0 0.0.0.255 10.0.0.0 0.0.0.255
! Create crypto map that will use transform set "myset", use peer
10.0.1.1
! to establish SAs and encapsulate traffic and use access-list
101 to
! match traffic that should be encrypted
crypto map mymap 10 ipsec-isakmp
  set peer 10.0.1.1
  set transform-set myset
  set pfs group2
  match address 101
  exit
! And finally apply crypto map to serial interface:
interface Serial 0
  crypto map mymap
```

```
  exit
```

4. Testing the IPsec tunnel
   - on **Hautespot** router we can see installed SAs

```
[admin@Hautespot] ip ipsec installed-sa> print
Flags: A - AH, E - ESP, P - pfs, M - manual
  0 E   spi=9437482 direction=out src-address=10.0.1.1
        dst-address=10.0.1.2 auth-algorithm=sha1 enc-
algorithm=des
        replay=4 state=mature
        auth-key="9cf2123b8b5add950e3e67b9eac79421d406aa09"
        enc-key="ffe7ec65b7a385c3" add-lifetime=24m/30m use-
lifetime=0s/0s
        lifebytes=0/0 current-addtime=jul/12/2002 16:13:21
        current-usetime=jul/12/2002 16:13:21 current-bytes=71896
  1 E   spi=319317260 direction=in src-address=10.0.1.2
        dst-address=10.0.1.1 auth-algorithm=sha1 enc-
algorithm=des
        replay=4 state=mature
        auth-key="7575f5624914dd312839694db2622a318030bc3b"
        enc-key="633593f809c9d6af" add-lifetime=24m/30m use-
lifetime=0s/0s
        lifebytes=0/0 current-addtime=jul/12/2002 16:13:21
        current-usetime=jul/12/2002 16:13:21 current-bytes=0
```

```
[admin@Hautespot] ip ipsec installed-sa>
```

   - on **CISCO** router

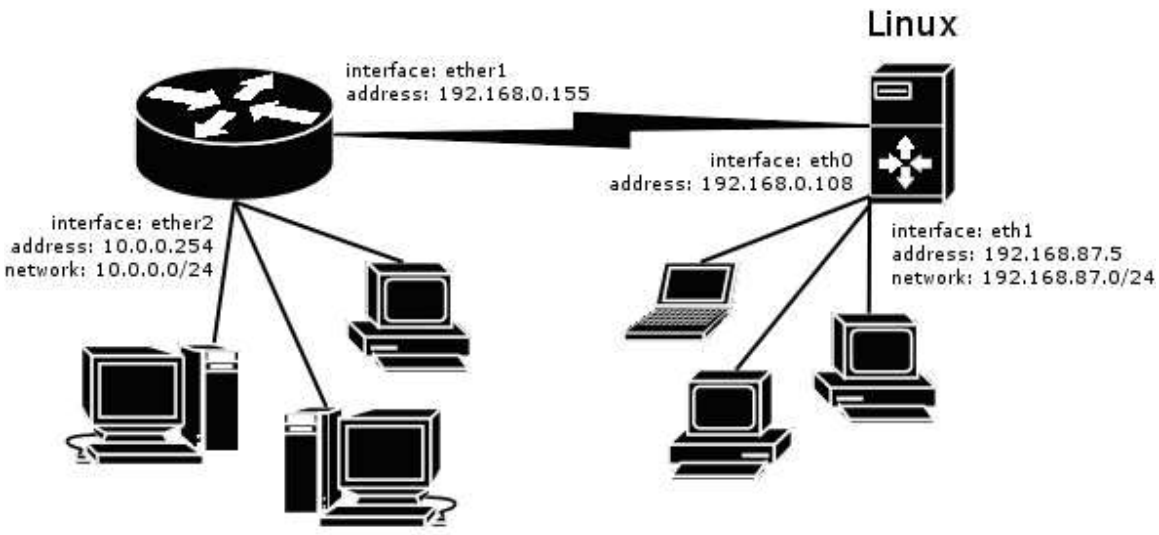
```
cisco# show interface Serial 0
interface: Serial1
    Crypto map tag: mymap, local addr. 10.0.1.2
   local  ident (addr/mask/prot/port):
(10.0.2.0/255.255.255.0/0/0)
   remote ident (addr/mask/prot/port):
(10.0.0.0/255.255.255.0/0/0)
   current_peer: 10.0.1.1
     PERMIT, flags={origin_is_acl,}
    #pkts encaps: 1810, #pkts encrypt: 1810, #pkts digest 1810
    #pkts decaps: 1861, #pkts decrypt: 1861, #pkts verify 1861
    #pkts compressed: 0, #pkts decompressed: 0
    #pkts not compressed: 0, #pkts compr. failed: 0, #pkts
decompress failed: 0
    #send errors 0, #recv errors 0
     local crypto endpt.: 10.0.1.2, remote crypto endpt.:
10.0.1.1
     path mtu 1500, media mtu 1500
     current outbound spi: 1308650C
     inbound esp sas:
      spi: 0x90012A(9437482)
        transform: esp-des esp-sha-hmac ,
        in use settings ={Tunnel, }
        slot: 0, conn id: 2000, flow_id: 1, crypto map: mymap
        sa timing: remaining key lifetime (k/sec): (4607891/1034)
        IV size: 8 bytes
        replay detection support: Y
```

```
o       inbound ah sas:
o       inbound pcp sas:
o       outbound esp sas:
o        spi: 0x1308650C(319317260)
o          transform: esp-des esp-sha-hmac ,
o          in use settings ={Tunnel, }
o          slot: 0, conn id: 2001, flow_id: 2, crypto map: mymap
o          sa timing: remaining key lifetime (k/sec): (4607893/1034)
o          IV size: 8 bytes
o          replay detection support: Y
o       outbound ah sas:
```

```
      outbound pcp sas:
```

Hautespot Router and Linux FreeS/WAN

In the test scenario we have 2 private networks: 10.0.0.0/24 connected to the MT and 192.168.87.0/24 connected to Linux. MT and Linux are connected together over the "public" network 192.168.0.0/24:



- FreeS/WAN configuration:

```
- config setup
-      interfaces="ipsec0=eth0"
-      klipsdebug=none
-      plutodebug=all
-      plutoload=%search
-      plutostart=%search
-      uniqueids=yes
-
- conn %default
```

```
    keyingtries=0
    disablearrivalcheck=no
    authby=rsasig

conn mt
    left=192.168.0.108
    leftsubnet=192.168.87.0/24
    right=192.168.0.155
    rightsubnet=10.0.0.0/24
    authby=secret
    pfs=no
```

```
    auto=add
```

- **ipsec.secrets** config file:

```
192.168.0.108 192.168.0.155 : PSK "gvejimezyfopmekun"
```

- Hautespot Router configuration:

```
[admin@Hautespot] > /ip ipsec peer add address=192.168.0.108 \
\... secret="gvejimezyfopmekun" hash-algorithm=md5  enc-algorithm=3des
\
\... dh-group=modp1024 lifetime=28800s

[admin@Hautespot] > /ip ipsec proposal auth-algorithms=md5 \
\... enc-algorithms=3des pfs-group=none

[admin@Hautespot] > /ip ipsec policy add sa-src-address=192.168.0.155 \
\... sa-dst-address=192.168.0.108 src-address=10.0.0.0/24 \
```

```
\... dst-address=192.168.87.0/24 tunnel=yes
```

HauteRouterOS IP Telephony

Document revision: 2.2 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

The HauteRouterOS IP Telephony feature enables Voice over IP (VoIP) communications using routers equipped with the following voice port hardware:

- Quicknet LineJACK or PhoneJACK analog telephony cards
- ISDN cards
- Voicetronix OpenLine4 (was V4PCI) - 4 analog telephone lines cards
- Zaptel Wildcard X100P IP telephony card - 1 analog telephone line

Specifications

Packages required: ***telephony***
License required: *Level1*
Submenu level: ***/ip telephony***
Standards and Technologies: RTP
Hardware usage: *Pentium MMX level processor recommended*

Description

IP telephony, known as Voice over IP (VoIP), is the transmission of telephone calls over a data network like one of the many networks that make up the Internet. There are four ways that you might talk to someone using VoIP:

- Computer-to-computer - This is certainly the easiest way to use VoIP, and you don't have to pay for long-distance calls.
- Computer-to-telephone - This method allows you to call anyone (who has a phone) from your computer. Like computer-to-computer calling, it requires a software client. The software is typically free, but the calls may have a small per-minute charge.
- Telephone-to-computer - Allows a standard telephone user to initiate a call to a computer user.
- Telephone-to-telephone - Through the use of gateways, you can connect directly with any other standard telephone in the world.

Suppoted hardware:

- Quicknet Technologies cards:
  - Internet PhoneJACK (ISA or PCI) for connecting an analog telephone (FXS port)
  - Internet LineJACK (ISA) for connecting an analog telephone line (FXO port) or a telephone (FXS port)
- ISDN client cards (PCI) for connecting an ISDN line. See Device Driver List for the list of supported PCI ISDN cards
- Voicetronix OpenLine4 card for connecting four (4) analog telephone lines (FXO ports)
- Zaptel Wildcard X100P IP telephony card (from Linux Support Services) for connecting one analog telephone line (FXO port)

Supported standards:

- Hautespot RouterOS supports IP Telephony in compliance with the International Telecommunications Union - Telecommunications (ITU-T) specification H.323v4. H.323 is a specification for transmitting multimedia (voice, video, and data) across an IP network. H.323v4 includes: H.245, H.225, Q.931, H.450.1, RTP(real-time protocol)
- The followong audio codecs are supported: **G.711** (the 64 kbps Pulse code modulation (PCM) voice coding), **G.723.1** (the 6.3 kbps compression technique that can be used for compressing audio signal at very low bit rate), **GSM-06.10** (the 13.2 kbps coding), **LPC-10** (the 2.5 kbps coding), **G.729** and **G.729a** (the 8 kbps CS-ACELP software coding), **G.728** (16 kbps coding technique, supported only on Quicknet LineJACK cards)

In PSTN lines there is a known delay of the signal caused by switching and signal compressing devices of the telephone network (so, it depends on the distance between the peers), which is generally rather low. The delay is also present in IP networks. The main difference between a PSTN and an IP network is that in IP networks that delay is more random. The actual packet delay may vary in order of magnutude in congested networks (if a network becomes congested, some packets may even be lost). Also packet reordering may take place. To prevent signal loss, caused by random jitter of IP networks and packet reordering, to corrupt audio signal, a jitter buffer is present in IP telephony devices. The jitter buffer is delaying the actual playback of a received packet forming The larger the jitter buffer, the larger the total delay, but fewer packets get lost due to timeout.

The total delay from the moment of recording the voice signal till its playback is the sum of following three delay times:

- delay time at the recording point (approx. 38ms)
- delay time of the IP network (1..5ms and up)
- delay time at the playback point (the jitter delay)

Notes

Each installed Quicknet card requires IO memory range in the following sequence: the first card occupies addresses 0x300-0x31f, the second card 0x320-0x33f, the third 0x340-0x35f, and so on. Make sure there is no conflict in these ranges with other devices, e.g., network interface cards, etc.

Use the telephony logging feature to debug your setup.

Additional Resources

IP Telephony Online

General Voice port settings

Submenu level: ***/ip telephony voice-port***

Description

This submenu is used for managing all IP telephony voice ports (linejack, phonejack, isdn, voip, voicetronix, zaptel)

Property Description

name (*name*) - assigned name of the voice port
type (*read-only:* phonejack | linejack | phonejack-lite | phonejack-pci | voip | isdn | voicetronix | zaptel) - type of the installed telephony voice port:
phonejack - Quicknet PhoneJACK (ISA)
linejack - Quicknet LineJACK (ISA)
phonejack-lite - Quicknet PhoneJACK Lite Linux Edition (ISA)
phonejack-pci - Quicknet PhoneJACK (PCI)
voip - generic Voice over IP port
isdn - ISDN cards
voicetronix - Voicetronix OpenLine4
zaptel - Zaptel Wildcard X100P
autodial (*integer*; default: **""**) - number to be dialed automatically, if call is coming in from this voice port

Notes

If **autodial** does not exactly match an item in **/ip telephony numbers**, there can be two possibilities:

- if **autodial** is incomplete, rest of the number is asked (local voice port) or incoming call is denied (VoIP)
- if **autodial** is invalid, line is hung up (PSTN line), busy tone is played (POTS) or incoming call is denied (VoIP)

Voicetronix Voice Ports

Submenu level: ***/ip telephony voice-port voicetronix***

Property Description

name (*name*) - name given by the user or the default one
autodial (*integer*; default: **""**) - phone number which will be dialed immediately after the handset has been lifted. If this number is incomplete, then the remaining part has to be dialed on the dial-pad. If the number is incorrect, the line is hung up. If the number is correct, then the appropriate number is dialed (the direct-call mode is used - the line is picked up only after the remote party answers the call)
playback-volume (*integer*: -48..48; default: **0**) - playback volume in dB
0 - 0dB meand no change to signal level
record-volume (*integer*: -48..48; default: **0**) - record volume in dB
0 - 0dB meand no change to signal level

region (*name*; default: **us**) - regional setting for the voice port. This setting is used for setting the parameters of PSTN line, as well as for detecting and generating the tones
agc-on-playback (yes | no; default: **no**) - automatic gain control on playback (can not be used together with hardware voice codecs)
agc-on-record (yes | no; default: **no**) - automatic gain control on record (can not be used together with hardware voice codecs)
detect-cpt (yes | no; default: **no**) - automatically detect call progress tones
balance-registers (*integer*: 0..255; default: **199**) - registers which depend on telephone line impedance. Can be adjusted to get best echo cancellation. Should be changed only if echo cancellation on voicetronix card does not work good enough. Echo cancellation problems can imply DTMF and busy-tone detection failures. The value has to be in format **bal1[,bal3[,bal2]]**, where **bal1**, **bal2**, **bal3** - balance registers. **bal1** has to be in interval 192..248 (0xC0..0xF8). The others should be in interval 0..255 (0x00..0xFF)
balance-status (*read-only: integer*; default: **unknown**) - shows quality of hardware echo cancellation in dB
loop-drop-detection (yes | no; default: **yes**) - automatically clear call when loop drop is detected

Command Description

test-balance - current **balance-registers** value is tested once. Result is placed in **balance-status** parameter. Balance can be tested only when line is off-hook. It won't work if line is on-hook or there is an established connection
Input Parameters
unnamed (*name*) - port name to test balance of
find-best-balance - series of **test-balance** is executed with different **balance-registers** values. During the tests **balance-registers** are updated to the best values found
Input Parameters
unnamed (*name*) - port name to find best balance of
clear-call - terminate a current call established with the specified voice port
Input Parameters
unnamed (*name*) - port name to clear call with
show-stats - show voice port statistics
Input Parameters
unnamed (*name*) - port name show statistics of
Return Values
round-trip-delay (*time*) - maximal time of packet round trip
packets-sent (*integer*) - number of packets sent by this card (these packets are digitalized input of the voice port)
bytes-sent (*integer*) - number of bytes sent by this card (these packets are digitalized input of the voice port)
sent-time (*text*) - minimal/average/maximal intervals between packets sent
packets-received (*integer*) - number of packets received by this card (these packets form analog output of the voice port)
bytes-received (*integer*) - number of bytes received by this card (these packets form analog output of the voice port)
received-time (*text*) - minimal/average/maximal intervals between packets received
average-jitter-delay (*time*) - approximate delay time from the moment of receiving an audio packet from the IP network till it is played back over the telephony voice port. The value shown is never less than 30ms, although the actual delay time could be less. If the shown value is >40ms, then it is close (+/-1ms) to the actual delay time.

monitor - monitor status of the voice port
Input Parameters
unnamed (*name*) - port name to monitor
Return Values
status (on-hook | off-hook | ring | connection | busy) - current state of the port:
- **on-hook** - the handset is on-hook, no activity
- **off-hook** - the handset is off-hook, the number is being dialed
- **ring** - call in progress, direction of the call is shown by the **direction** property
- **connection** - the connection has been established
- **busy** - the connection has been terminated, the handset is still off-hook

directiopn (ip-to-port | port-to-ip) - direction of the call
- **ip-to-port** - call from the IP network to the voice card
- **port-to-ip** - call from the voice card to an IP address

phone-number (*integer*) - the phone number being dialed
remote-party-name (*text*) - name and IP address of the remote party
codec (*name*) - CODEC used for the audio connection
duration (*time*) - duration of the phone call

Notes

As some Voicetronix cards fail to detect loop drop correctly, with **loop-drop-detection** you can manage whether loop drop detection feature is enabled. The effect of not working loop-drop detection is call terminated at once when connection is established.

Some tips for testing balance registers:

- test is sensitive to noise from the phone, so it's recommended to cover mouth piece during it;
- **find-best-balance** can be interrupted by **clear-call** command;
- once best **balance-registers** value is known, it can be set manually to this best value for all voicetronix voice ports, which will use the same telephone line.

LineJack Voice Ports

Submenu level: ***/ip telephony voice-port linejack***

Property Description

name (*name*) - name given by the user or the default one
autodial (*integer*; default: **""**) - phone number which will be dialed immediately after the handset has been lifted. If this number is incomplete, then the remaining part has to be dialed on the dial-pad. If the number is incorrect, the line is hung up (FXO "line" port) or busy tone is played (FXS "phone" port). If the number is correct, then the appropriate number is dialed. If it is an incomming call from the PSTN line, then the direct-call mode is used - the line is picked up only after the remote party answers the call
playback-volume (*integer*: -48..48; default: **0**) - playback volume in dB
0 - 0dB meand no change to signal level

record-volume (*integer*: -48..48; default: **0**) - record volume in dB
0 - 0dB meand no change to signal level
ring-cadence (*text*) - a 16-symbol ring cadence for the phone, each symbol lasts 0.5 seconds, **+** means ringing, **-** means no ringing
region (*name*; default: **us**) - regional setting for the voice port. This setting is used for setting the parameters of PSTN line, as well as for detecting and generating the tones
aec (yes | no) - whether echo detection and cancellation is enabled
aec-tail-length (short | medium | long; default: **short**) - size of the buffer of echo detection
aec-nlp-threshold (off | low | medium | high; default: **low**) - level of cancellation of silent sounds
aec-attenuation-scaling (*integer*: 0..10; default: **4**) - factor of additional echo attenuation
aec-attenuation-boost (*integer*: 0..90; default: **0**) - level of additional echo attenuation
software-aec (yes | no) - software echo canceller (experimental, for most of the cards)
agc-on-playback (yes | no; default: **no**) - automatic gain control on playback (can not be used together with hardware voice codecs)
agc-on-record (yes | no; default: **no**) - automatic gain control on record (can not be used together with hardware voice codecs)
detect-cpt (yes | no; default: **no**) - automatically detect call progress tones

Command Description

blink - blink the LEDs of the specified voice port for five seconds after it is invoked. This command can be used to locate the respective card from several linejack cards
Input Parameters
unnamed (*name*) - card name to blink the LED of
clear-call - terminate a current call established with the specified voice port
Input Parameters
unnamed (*name*) - port name to clear call with
show-stats - show voice port statistics
Input Parameters
unnamed (*name*) - port name show statistics of
Return Values
round-trip-delay (*time*) - maximal time of packet round trip
packets-sent (*integer*) - number of packets sent by this card (these packets are digitalized input of the voice port)
bytes-sent (*integer*) - number of bytes sent by this card (these packets are digitalized input of the voice port)
sent-time (*text*) - minimal/average/maximal intervals between packets sent
packets-received (*integer*) - number of packets received by this card (these packets form analog output of the voice port)
bytes-sent (*integer*) - number of bytes received by this card (these packets form analog output of the voice port)
sent-time (*text*) - minimal/average/maximal intervals between packets received
average-jitter-delay (*time*) - approximate delay time from the moment of receiving an audio packet from the IP network till it is played back over the telephony voice port. The value shown is never less than 30ms, although the actual delay time could be less. If the shown value is >40ms, then it is close (+/-1ms) to the actual delay time.
monitor - monitor status of the voice port

Input Parameters
unnamed (*name*) - port name to monitor
Return Values
status (on-hook | off-hook | ring | connection | busy) - current state of the port:
- **on-hook** - the handset is on-hook, no activity
- **off-hook** - the handset is off-hook, the number is being dialed
- **ring** - call in progress, direction of the call is shown by the **direction** property
- **connection** - the connection has been established
- **busy** - the connection has been terminated, the handset is still off-hook

port (phone | line) - the active port of the card
- **phone** - telephone connected to the card (POTS FXS port)
- **line** - line connected to the card (PSTN FXO port)

directiopn (ip-to-port | port-to-ip) - direction of the call
- **ip-to-port** - call from the IP network to the voice card
- **port-to-ip** - call from the voice card to an IP address

line-status (plugged | unplugged) - state of the PSTN line
- **plugged** - the telephone line is connected to the PSTN port of the card
- **unplugged** - there is no working line connected to the PSTN port of the card

phone-number (*integer*) - the phone number being dialed
remote-party-name (*text*) - name and IP address of the remote party
codec (*name*) - CODEC used for the audio connection
duration (*time*) - duration of the phone call

Notes

When telephone line is connected to the 'line' port, green LED next to the port should be lit in some seconds. If telephone line disappear, the LED next to the 'line' port will change its state to red in an hour or when the line is activated (i.e. when somebody calls to/from it). When telephone line is plugged in the 'phone' port before the router is turned on, red LED next to the port will be lit. WARNING: do not plug telephone line into the 'phone' port when the router is running and green LED next to the port is lit - this might damage the card. The status of the 'phone' port is only detected on system startup.

PhoneJack Voice Ports

Submenu level: ***/ip telephony voice-port phonejack***

Property Description

name (*name*) - name given by the user or the default one
type (*read-only:* phonejack | phonejack-lite | phonejack-pci) - type of the card
autodial (*integer*; default: **""**) - phone number which will be dialed immediately after the handset has been lifted. If this number is incomplete, then the remaining part has to be dialed on the dial-pad. If the number is incorrect, busy tone is played. If the number is correct, then the appropriate number is dialed
playback-volume (*integer*: -48..48; default: **0**) - playback volume in dB
0 - 0dB meand no change to signal level
record-volume (*integer*: -48..48; default: **0**) - record volume in dB
0 - 0dB meand no change to signal level
ring-cadence (*text*) - a 16-symbol ring cadence for the phone, each symbol lasts 0.5 seconds, **+** means ringing, **-** means no ringing

region (*name*; default: **us**) - regional setting for the voice port. This setting is used for generating the dial tones
aec (yes | no) - wheteher echo detection and cancellation is enabled
aec-tail-length (short | medium | long; default: **short**) - size of the buffer of echo detection
aec-nlp-threshold (off | low | medium | high; default: **low**) - level of cancellation of silent sounds
aec-attenuation-scaling (*integer*: 0..10; default: **4**) - factor of additional echo attenuation
aec-attenuation-boost (*integer*: 0..90; default: **0**) - level of additional echo attenuation
software-aec (yes | no) - software echo canceller (experimental, for most of the cards)
agc-on-playback (yes | no; default: **no**) - automatic gain control on playback (can not be used together with hardware voice codecs)
agc-on-record (yes | no; default: **no**) - automatic gain control on record (can not be used together with hardware voice codecs)
detect-cpt (yes | no; default: **no**) - automatically detect call progress tones

Command Description

clear-call - terminate a current call established with the specified voice port
Input Parameters
unnamed (*name*) - port name to clear call with
show-stats - show voice port statistics
Input Parameters
unnamed (*name*) - port name show statistics of
Return Values
round-trip-delay (*time*) - maximal time of packet round trip
packets-sent (*integer*) - number of packets sent by this card (these packets are digitalized input of the voice port)
bytes-sent (*integer*) - number of bytes sent by this card (these packets are digitalized input of the voice port)
sent-time (*text*) - minimal/average/maximal intervals between packets sent
packets-received (*integer*) - number of packets received by this card (these packets form analog output of the voice port)
bytes-sent (*integer*) - number of bytes received by this card (these packets form analog output of the voice port)
sent-time (*text*) - minimal/average/maximal intervals between packets received
average-jitter-delay (*time*) - approximate delay time from the moment of receiving an audio packet from the IP network till it is played back over the telephony voice port. The value shown is never less than 30ms, although the actual delay time could be less. If the shown value is >40ms, then it is close (+/-1ms) to the actual delay time.
monitor - monitor status of the voice port
Input Parameters
unnamed (*name*) - port name to monitor
Return Values
status (on-hook | off-hook | ring | connection | busy) - current state of the port:
- **on-hook** - the handset is on-hook, no activity
- **off-hook** - the handset is off-hook, the number is being dialed
- **ring** - call in progress, direction of the call is shown by the **direction** property
- **connection** - the connection has been established
- **busy** - the connection has been terminated, the handset is still off-hook

port (phone | line) - the active port of the card
- **phone** - telephone connected to the card (POTS FXS port)

- **line** - line connected to the card (PSTN FXO port)

directiopn (ip-to-port | port-to-ip) - direction of the call
- **ip-to-port** - call from the IP network to the voice card
- **port-to-ip** - call from the voice card to an IP address

line-status (plugged | unplugged) - state of the PSTN line
- **plugged** - the telephone line is connected to the PSTN port of the card
- **unplugged** - there is no working line connected to the PSTN port of the card

phone-number (*integer*) - the phone number being dialed
remote-party-name (*text*) - name and IP address of the remote party
codec (*name*) - CODEC used for the audio connection
duration (*time*) - duration of the phone call

Zaptel Voice Ports

Submenu level: ***/ip telephony voice-port zaptel***

Property Description

name (*name*) - name given by the user or the default one
autodial (*integer*; default: **""**) - phone number which will be dialed immediately after the handset has been lifted. If this number is incomplete, then the remaining part has to be dialed on the dial-pad. If the number is incorrect, the line is hung up. If the number is correct, then the appropriate number is dialed (the direct-call mode is used - the line is picked up only after the remote party answers the call)
playback-volume (*integer*: -48..48; default: **0**) - playback volume in dB
0 - 0dB meand no change to signal level
record-volume (*integer*: -48..48; default: **0**) - record volume in dB
0 - 0dB meand no change to signal level
region (*name*; default: **us**) - regional setting for the voice port. This setting is used for setting the parameters of PSTN line, as well as for detecting and generating the tones
aec (yes | no) - whetеher echo detection and cancellation is enabled
aec-tail-length (short | medium | long; default: **short**) - size of the buffer of echo detection
aec-nlp-threshold (off | low | medium | high; default: **low**) - level of cancellation of silent sounds
aec-attenuation-scaling (*integer*: 0..10; default: **4**) - factor of additional echo attenuation
aec-attenuation-boost (*integer*: 0..90; default: **0**) - level of additional echo attenuation
software-aec (yes | no) - software echo canceller (experimental, for most of the cards)
agc-on-playback (yes | no; default: **no**) - automatic gain control on playback (can not be used together with hardware voice codecs)
agc-on-record (yes | no; default: **no**) - automatic gain control on record (can not be used together with hardware voice codecs)
detect-cpt (yes | no; default: **no**) - automatically detect call progress tones

Command Description

clear-call - terminate a current call established with the specified voice port
Input Parameters
unnamed (*name*) - port name to clear call with
show-stats - show voice port statistics
Input Parameters
unnamed (*name*) - port name show statistics of
Return Values
round-trip-delay (*time*) - maximal time of packet round trip
packets-sent (*integer*) - number of packets sent by this card (these packets are digitalized input of the voice port)
bytes-sent (*integer*) - number of bytes sent by this card (these packets are digitalized input of the voice port)
sent-time (*text*) - minimal/average/maximal intervals between packets sent
packets-received (*integer*) - number of packets received by this card (these packets form analog output of the voice port)
bytes-sent (*integer*) - number of bytes received by this card (these packets form analog output of the voice port)
sent-time (*text*) - minimal/average/maximal intervals between packets received
average-jitter-delay (*time*) - approximate delay time from the moment of receiving an audio packet from the IP network till it is played back over the telephony voice port. The value shown is never less than 30ms, although the actual delay time could be less. If the shown value is >40ms, then it is close (+/-1ms) to the actual delay time.
monitor - monitor status of the voice port
Input Parameters
unnamed (*name*) - port name to monitor
Return Values
status (on-hook | off-hook | ring | connection | busy) - current state of the port:
- **on-hook** - the handset is on-hook, no activity
- **off-hook** - the handset is off-hook, the number is being dialed
- **ring** - call in progress, direction of the call is shown by the **direction** property
- **connection** - the connection has been established
- **busy** - the connection has been terminated, the handset is still off-hook

directiopn (ip-to-port | port-to-ip) - direction of the call
- **ip-to-port** - call from the IP network to the voice card
- **port-to-ip** - call from the voice card to an IP address

line-status (plugged | unplugged) - state of the PSTN line
- **plugged** - the telephone line is connected to the PSTN port of the card
- **unplugged** - there is no working line connected to the PSTN port of the card

phone-number (*integer*) - the phone number being dialed
remote-party-name (*text*) - name and IP address of the remote party
codec (*name*) - CODEC used for the audio connection
duration (*time*) - duration of the phone call

ISDN Voice Ports

Submenu level: ***/ip telephony voice-port isdn***

Property Description

name (*name*) - name given by the user or the default one
msn (*integer*) - telephone number of the ISDN voice port (ISDN MSN number)
lmsn (*text*) - msn pattern to listen on. It determines which calls from the ISDN line this voice port should answer. If left empty, **msn** is used
autodial (*integer*; default: **""**) - phone number which will be dialed immediately on each incoming ISDN call. If this number contains 'm', then it will be replaced by

originally called (ISDN) telephone number. If this number is incomplete, then the remaining part has to be dialed by the caller. If the number is incorrect, call is refused. If the number is correct, then the appropriate number is dialed. For that direct-call mode is used - the line is picked up only after the remote party answers the call
playback-volume (*integer*: -48..48; default: **0**) - playback volume in dB
0 - 0dB meand no change to signal level
record-volume (*integer*: -48..48; default: **0**) - record volume in dB
0 - 0dB meand no change to signal level
region (*name*; default: **us**) - regional setting for the voice port. This setting is used for setting the parameters of PSTN line, as well as for detecting and generating the tones
aec (yes | no) - whetеher echo detection and cancellation is enabled
aec-tail-length (short | medium | long; default: **short**) - size of the buffer of echo detection
software-aec (yes | no) - software echo canceller (experimental, for most of the cards)
agc-on-playback (yes | no; default: **no**) - automatic gain control on playback (can not be used together with hardware voice codecs)
agc-on-record (yes | no; default: **no**) - automatic gain control on record (can not be used together with hardware voice codecs)

Command Description

clear-call - terminate a current call established with the specified voice port
Input Parameters
unnamed (*name*) - port name to clear call with
show-stats - show voice port statistics
Input Parameters
unnamed (*name*) - port name show statistics of
Return Values
round-trip-delay (*time*) - maximal time of packet round trip
packets-sent (*integer*) - number of packets sent by this card (these packets are input of the voice port)
bytes-sent (*integer*) - number of bytes sent by this card (these packets are input of the voice port)
sent-time (*text*) - minimal/average/maximal intervals between packets sent
packets-received (*integer*) - number of packets received by this card (these packets form output of the voice port)
bytes-sent (*integer*) - number of bytes received by this card (these packets form output of the voice port)
sent-time (*text*) - minimal/average/maximal intervals between packets received
average-jitter-delay (*time*) - approximate delay time from the moment of receiving an audio packet from the IP network till it is played back over the telephony voice port. The value shown is never less than 30ms, although the actual delay time could be less. If the shown value is >40ms, then it is close (+/-1ms) to the actual delay time.
monitor - monitor status of the voice port
Input Parameters
unnamed (*name*) - port name to monitor
Return Values
status (on-hook | off-hook | ring | connection | busy) - current state of the port:
- **on-hook** - the handset is on-hook, no activity
- **off-hook** - the handset is off-hook, the number is being dialed
- **ring** - call in progress, direction of the call is shown by the **direction** property
- **connection** - the connection has been established

- **busy** - the connection has been terminated, the handset is still off-hook

directiopn (ip-to-port | port-to-ip) - direction of the call
- **ip-to-port** - call from the IP network to the voice card
- **port-to-ip** - call from the voice card to an IP address

phone-number (*integer*) - the phone number being dialed
remote-party-name (*text*) - name and IP address of the remote party
codec (*name*) - CODEC used for the audio connection
duration (*time*) - duration of the phone call

Notes

In contrary to analog voice ports phonejack, linejack, voicetronix, zaptel), which are as many as the number of cards installed, the isdn ports can be added as many as desired.

There is a possibility to enter some special symbols in **lmsn** property. Meaning of the special symbols:

- **;** - separates pattern entries (more than one pattern can be specified this way)
- **?** - matches one character
- ***** - matches zero or more characters
- **[ ]** - matches any single character from the set in brackets
- **[^]** - matches any single character not from the set in brackets

Voice Port for Voice over IP (voip)

Submenu level: ***/ip telephony voice-port voip***

Description

The voip voice ports are virtual ports, which designate a voip channel to another host over the IP network. You must have at least one voip voice port to be able to make calls to other H.323 devices over IP network.

Property Description

name (*name*) - name given by the user or the default one
remote-address (*IP address*; default: **0.0.0.0**) - IP address of the remote party (IP telephone or gateway) associated with this voice port. If the call has to be performed through this voice port, then the specified IP address is called. If there is an incoming call from the specified IP address, then the parameters of this voice port are used. If there is an incoming call from an IP address, which is not specified in any of the voip voice port records, then the default record is used. If there is no default record, then default values are used
0.0.0.0 - the record with this IP address will specify the default values for an incomming call
autodial (*integer*) - phone number which will be added in front of the telephone number received over the IP network. In most cases it should be blank
jitter-buffer (*time*: 0..1000ms; default: **100ms**) - size of the jitter buffer

0 - the size of it is adjusted automatically during the conversation, to keep amount of lost packets under 1%
silence-detection (yes | no; default: **no**) - whether silence is detected and no audio data is sent over the IP network during the silence period
prefered-codec (*name*; default: **none**) - the preferred codec to be used for this voip voice port. If possible, the specified codec will be used
none - there is no preferred codec defined for this port, so whichever codec advised by the remote peer will be used (if it is supported)
fast-start (yes | no; default: **yes**) - allow or disallow the fast start. The fast start allows establishing the audio connection in a shorter time. However, not all H.323 endpoints support this feature. Therefore, it should be turned off, if there are problems to establish telephony connection using the fast start mode

Numbers

Description

This is the so-called "routing table" for voice calls. This table assigns numbers to the voice ports.The main function of the numbers routing table is to determine:

- to which voice port route the call
- what number to send over to the remote party

Property Description

dst-pattern (*integer*) - pattern of the telephone number. Symbol '.' designate any digit, symbol '_' (only as the last one) designate any symbols (i.e. any number of characters can follow, ended with '#' button)
voice-port (*name*) - voice port to be used when calling the specified telephone number
prefix (*integer*) - prefix, which will be used to substitute the known part of the **dst-pattern**, i.e., the part containing digits. The **dst-pattern** argument is used to determine which voice port to be used, whereas the **prefix** argument designates the number to dial over the voice port (be sent over to the remote party). If the remote party is an IP telephony gateway, then the number will be used for making the call

Notes

More than one entry can be added with exactly the same **dst-pattern**. If first one of them is already busy, next one with the same **dst-pattern** is used. Telephony number entries can be moved, to select desired order.

Example

Let us consider the following example for the number table:

```
[admin@Hautespot] ip telephony numbers> print
```

```
Flags: I - invalid, X - disabled, D - dynamic, R - registered
  #     DST-PATTERN                    VOICE-PORT PREFIX
  0     12345                          XX
  1     1111.                          YY
  2     22...                          ZZ         333
  3     ...                            QQ         55

[admin@Hautespot] ip telephony numbers>
```

We will analyze the Number Received (nr) - number dialed at the telephone, or received over the line, the Voice Port (vp) - voice port to be used for the call, and the Number to Call (nc) - number to be called over the Voice Port.

- If nr=55555, it does not match any of the destination patterns, therefore it is rejected
- If nr=123456, it does not match any of the destination patterns, therefore it is rejected
- If nr=1234, it does not match any of the destination patterns (incomplete for record #0), therefore it is rejected
- If nr=12345, it matches the record #0, therefore number "" is dialed over the voice port XX
- If nr=11111, it matches the record #1, therefore number "1" is dialed over the voice port YY
- If nr=22987, it matches the record #2, therefore number "333987" is dialed over the voice port ZZ
- If nr=22000, it matches the record #2, therefore number "333000" is dialed over the voice port ZZ
- If nr=444, it matches the record #3, therefore number "55444" is dialed over the voice port QQ

Let us add a few more records:

```
[admin@Hautespot] ip telephony numbers> print
Flags: I - invalid, X - disabled, D - dynamic, R - registered
  #     DST-PATTERN                    VOICE-PORT PREFIX
  0     12345                          XX
  1     1111.                          YY
  2     22...                          ZZ         333
  3     ...                            QQ         55
  4     222                            KK         44444
  5     3..                            LL         553

[admin@Hautespot] ip telephony numbers>
```

- If nr=222 => the best match is the record #4 => nc=44444, vp=KK (note: the 'best match' means that it has the most coinciding digits between the nr and destination pattern).
- If nr=221 => incomplete record #2 => call is rejected

- If nr=321 => the best match is the record #5 => nc=55321, vp=LL
- If nr=421 => matches the record #3 => nc=55421, vp=QQ
- If nr=335 => the best match is the record #5 => nc=55321, vp=LL

Let us add a few more records:

```
[admin@Hautespot] ip telephony numbers> print
Flags: I - invalid, X - disabled, D - dynamic, R - registered
Flags: I - invalid, X - disabled, D - dynamic, R - registered
  #     DST-PATTERN                    VOICE-PORT PREFIX
  0     12345                          XX
  1     1111.                          YY
  2     22...                          ZZ         333
  3     ...                            QQ         55
  4     222                            KK         44444
  5     3..                            LL         553
  6     33...                          MM         33
  7     11.                            NN         7711

[admin@Hautespot] ip telephony numbers>
```

- If nr=335 => incomplete record #6 => the call is rejected. The nr=335 fits perfectly both the record #3 and #5. The #5 is chosen as the 'best match' candidate at the moment. Furthermore, there is record #6, which has two matching digits (more than for #3 or #5). Therefore the #6 is chosen as the 'best match'. However, the record #6 requires five digits, but the nr has only three. Two digits are missing, therefore the number is incomplete. Two additional digits would be needed to be entered on the dialpad. If the number is sent over from the network, it is rejected.
- If nr=325 => matches the record #5 => nc=55325, vp=LL
- If nr=33123 => matches the record #6 => nc=33123, vp=MM
- If nr=123 => incomplete record #0 => call is rejected
- If nr=111 => incomplete record #1 => call is rejected
- If nr=112 => matches the record #7 => nc=77112, vp=NN
- If nr=121 => matches the record #3 => nc=55121, vp=QQ

It is impossible to add the following records:

```
  #     DST-PATTERN                    VOICE-PORT PREFIX
                                                  reason:
        11                             DD         conflict with record #
1
                                                  and # 7
        11..                           DD         conflict with record #
7
        111                            DD         conflict with record #
1
        22.                            DD         conflict with record #
2
        .....                          DD         conflict with record #
3
```

Regional Settings

Submenu level: ***/ip telephony region***

Description

Regional settings are used to adjust the voice port properties to the PSTN system or the PBX. For example, to detect hang-up from line, there has to be correct regional setting (correct busy-tone-frequency and busy-tone-cadence). Without that, detect-cpt parameter the voice port has to be enabled.

Property Description

name (*name*) - name of the regional setting
busy-tone-cadence (*integer*: 0..30000; default: **500,500**) - busy tone cadence in ms
0 - end of cadence
busy-tone-frequency (*integer*: 20..2000*integer*: -24..6; default: **440x0**) - frequency and volume gain of busy tone, Hz x dB
data-access-arrangement (australia | france | germany | japan | uk | us; default: **us**) - ring voltage, impedance setting for line-jack card
dial-tone-frequency (*integer*: 20..2000*integer*: -24..6; default: **440x0**) - frequency and volume gain of dial tone, Hz x dB
dtmf-tone-cadence (*integer*: 0..30000; default: **180,60**) - Dual Tone Multi Frequency tone cadence in ms
0 - end of cadence
dtmf-tone-volume (*integer*: -24..6; default: **-3,-3**) - Dual Tone Multi Frequency tone volume in dB
ring-tone-cadence (*integer*: 0..30000; default: **1000,2000**) - Ring tone cadence in ms
0 - end of cadence
ring-tone-frequency (*integer*: 20..2000*integer*: -24..6; default: **440x0**) - frequency and volume gain of busy tone, Hz x dB

Notes

To generate a tone, frequency and cadence arguments are used. The dialtone always is continuous signal, therefore it does not have the cadence argument. In order to detect dialtone, it should be at least 100ms long.

There are 10 pre-defined regions, which can not be deleted (but may be changed)

Audio CODECs

Submenu level: ***/ip telephony codec***

Description

CODECs are listed according to their priority of use. The highest priority is at the top. CODECs can be enabled, disabled and moved within the list. When connecting with other H.323 systems, the protocol will negotiate the CODEC which both of them support according to the priority order.

The hardware codecs (/hw) are built-in CODECs supported by some cards.

The choice of the CODEC type is based on the throughput and speed of the network. Better audio quality can be achieved by using CODEC requiring higher network throughput. The highest audio quality can be achieved by using the G.711-uLaw CODEC requiring 64kb/s throughput for each direction of the call. It is used mostly within a LAN. The G.723.1 CODEC is the most popular one to be used for audio connections over the Internet. It requires only 6.3kb/s throughput for each direction of the call.

Example

```
[admin@Hautespot] ip telephony codec> print
Flags: X - disabled
  #    NAME
  0    G.723.1-6.3k/sw
  1    G.728-16k/hw
  2    G.711-ALaw-64k/hw
  3    G.711-uLaw-64k/hw
  4    G.711-uLaw-64k/sw
  5    G.711-ALaw-64k/sw
  6    G.729A-8k/sw
  7    GSM-06.10-13.2k/sw
  8    LPC-10-2.5k/sw
  9    G.723.1-6.3k/hw
 10    G.729-8k/sw
[admin@Hautespot] ip telephony codec>
```

AAA

Submenu level: ***/ip telephony aaa***

Description

AAA (Authentication Authorization Accounting) can be used to configure the RADIUS accounting feature.

The contents of the CDR (Call Detail Record) are as follows:

- **NAS-Identifier** - router name (from **/system identity print**)
- **NAS-IP-Address** - router's local IP address which the connection was established to (if exist)
- **NAS-Port-Type** - always **Async**
- **Event-Timestamp** - data and time of the event
- **Acct-Session-Time** - current connection duration (only in INTERIM-UPDATE and STOP records)
- **Acct-Output-Packets** - sent RTP (Real-Time Transport Protocol) packet count (only in INTERIM-UPDATE and STOP records)
- **Acct-Output-Packets** - sent RTP (Real-Time Transport Protocol) packet count (only in INTERIM-UPDATE and STOP records)
- **Acct-Input-Packets** - received RTP (Real-Time Transport Protocol) packet count (only in INTERIM-UPDATE and STOP records)
- **Acct-Output-Octets** - sent byte count (only in INTERIM-UPDATE and STOP records)
- **Acct-Input-Octets** - received byte count (only in INTERIM-UPDATE and STOP records)
- **Acct-Session-Id** - unique session participient ID
- **h323-disconnect-cause** - session disconnect reason (only in STOP records):
    - **0** - Local endpoint application cleared call
    - **1** - Local endpoint did not accept call
    - **2** - Local endpoint declined to answer call
    - **3** - Remote endpoint application cleared call
    - **4** - Remote endpoint refused call
    - **5** - Remote endpoint did not answer in required time
    - **6** - Remote endpoint stopped calling
    - **7** - Transport error cleared call
    - **8** - Transport connection failed to establish call
    - **9** - Gatekeeper has cleared call
    - **10** - Call failed as could not find user (in GK)
    - **11** - Call failed as could not get enough bandwidth
    - **12** - Could not find common capabilities
    - **13** - Call was forwarded using FACILITY message
    - **14** - Call failed a security check and was ended
    - **15** - Local endpoint busy
    - **16** - Local endpoint congested
    - **17** - Remote endpoint busy
    - **18** - Remote endpoint congested
    - **19** - Could not reach the remote party
    - **20** - The remote party is not running an endpoint
    - **21** - The remote party host off line
    - **22** - The remote failed temporarily app may retry
- **h323-disconnect-time** - session disconnect time (only in INTERIM-UPDATE and STOP records)
- **h323-connect-time** - session establish time (only in INTERIM-UPDATE and STOP records)
- **h323-gw-id** - name of gateway emitting message (should be equal to **NAS-Identifier**)
- **h323-call-type** - call leg type (should be **VoIP**)
- **h323-call-origin** - indicates origin of call relatively to the gateway (**answer** for calls from IP network, **originate** - to IP network)

- **h323-setup-time** - call setup time
- **h323-conf-id** - unique session ID
- **h323-remote-address** - the remote address of the session
- **NAS-Port-Id** - voice port ID
- **Acct-Status-Type** - record type (**START** when session is established; **STOP** when session is closed; **INTERIM-UPDATE (ALIVE)**session is alive). The time between the interim-update messages is defined by the **interim-update-interval** parameter (if it is set to **0s**, there will be no such messages)

Property Description

use-radius-accounting (yes | no; default: **no**) - whether to use radius accounting or not
interim-update (*integer*; default: **0**) - defines time interval between communications with the router. If this time will exceed, RADIUS server will assume that this connection is down. This value is suggested not to be less than 3 minutes
0 - no interim-update messages are sent at all

Notes

All the parameters, which names begin with **h323**, are CISCO vendor specific Radius attributes

Gatekeeper

Submenu level: ***/ip telephony gatekeeper***

Description

For each H.323 endpoint gatekeeper stores its telephone numbers. So, gatekeeper knows all telephone numbers for all registered endpoints. And it knows which telephone number is handled by which endpoint. Mapping between endpoints and their telephone numbers is the main functionality of gatekeepers.

If endpoint is registered to endpoint, it does not have to know every single endpoint and every single telephone number, which can be called. Instead, every time some number is dialed, endpoint asks gatekeeper for destination endpoint to call by providing called telephone number to it.

Hautespot IP telephony package includes a very simple gatekeeper. This gatekeeper can be activated by setting **gatekeeper** parameter to **local**. In this case the local endpoint automatically is registered to the local gatekeeper. And any other endpoint can register to this gatekeeper too.

Registered endpoints are added to the **/ip telephony voice-port voip** table. Those entries are marked as dynamic and can not be removed or changed. If there already

was an voip entry with the same IP address, it is marked as registred. Remote-address can not be changed for these entries too, but registered voip voice ports can be removed - they will stay as dynamic ones. If there already is a dynamic voip voice port and a static one with the same IP address is added, then instead of dynamic entry, registered will appear.

Dynamic entries disappear when corresponding endpoint unregisters itself from the gatekeeper. Registered entries are static and will stay even after that endpoint will be unregistered from this gatekeeper.

Registered telephone numbers are added to **/ip telephony numbers** table. Here is exactly the same idea behind dynamic and registered telephone numbers as it is with voip voice ports.

When an endpoint registers to the gatekeeper, it sends its own telephone numbers (aliases and prefixes) within this registration request. **/ip telephony numbers** entry is registered to the endpoint only if voice-port for that entry is local (not voip). If **dst-pattern** contains '.' or '_', it is sent as prefix, otherwise - as alias. The known part of the **dst-pattern** is sent as prefix. If there is no known part (**dst-pattern** is "_" or "...", for example), then this entry is not sent at all.

Property Description

gatekeeper (none | local | remote; default: **none**) - Gatekeeper type to use
none - don't use any gatekeeper at all
local - start and use local gatekeeper
remote - use some other gatekeeper
remote-address (*IP address*; default: **0.0.0.0**) - IP address of remote gatekeeper to use. If set to **0.0.0.0**, broadcast gatekeeper discovery is used
remote-id (*name*) - name of remote gatekeeper to use. If left empty, first available gatekeeper will be used. Name of locally started gatekeeper is the same as system identity
registered (*read-only:* yes | no) - shows whether local H.323 endpoint is registered to any gatekeeper
registered-with (*read-only: name*) - name of gatekeeper to which local H.323 endpoint is registered

Example

In most simple case with one phonejack card and some remote gatekeeper, configuration can be as follows:

```
[admin@Hautespot] ip telephony voice-port> print
Flags: X - disabled
  #   NAME                          TYPE            AUTODIAL
  0   phonejack1                phonejack
  1   voip1                         voip

[admin@Hautespot] ip telephony voice-port voip> print
Flags: X - disabled, D - dynamic, R - registered
```

```
  #    NAME      AUTODIAL REMOTE-ADDRESS  JITTER-BUFFER PREFERED-CODEC  SIL
FAS
  0    voip1              0.0.0.0         0s            none            no
yes

[admin@Hautespot] ip telephony numbers> print
Flags: I - invalid, X - disabled, D - dynamic, R - registered
  #     DST-PATTERN             VOICE-PORT              PREFIX
  0     11                      phonejack1
  1     _                       voip1

[admin@Hautespot] ip telephony gatekeeper> print
         gatekeeper: remote
          remote-id: ""
     remote-address: 10.0.0.98
         registered: yes
    registered-with: "Hautespot@10.0.0.98"
```

In this case this endpoint will register to gatkeeper with the IP address of 10.0.0.98 and telephone number 11. Every call to telephone number 11 will be transfered from gatekeeper to this endpoint. And this endpoint will route this call to phonejack1 voice port. On any other telephone number gatekeeper will be asked for real destination. From this endpoint it will be possible to call all the endpoints, which are registered to the same gatekeeper. If that gatekeeper has static entries about endpoints, which are not registered to gatekeeper, it still will be possible to call those endpoints by those statically defined telephone numbers at gatekeeper.

Example

For example, if numbers table is like this:

```
[admin@Hautespot] ip telephony numbers> print
Flags: I - invalid, X - disabled, D - dynamic, R - registered
  #     DST-PATTERN             VOICE-PORT              PREFIX
  0     1.                      phonejack1
  1     128                     voip1                   128
  2     78                      voip2                   78
  3     77                      phonejack1
  4     76                      phonejack1              55
  5     _                       voip1
```

then entries 0, 3 and 4 will be sent to the gatekeeper, others are voip voice ports and are ignored. Entry 0 will be sent as prefix 1, entry 3 - as alias 77, and entry 4 - as alias 76.

If IP address of local endpoint is 10.0.0.100, then gatekeeper voip and numbers tables will look as follows:

```
      [admin@Hautespot] ip telephony voice-port voip> print
Flags: X - disabled, D - dynamic, R - registered
  #    NAME       AUTODIAL REMOTE-ADDRESS  JITTER-BUFFER PREFERED-CODEC  SIL
FAS
  0    tst-2.5            10.0.0.101      0s            none            no
yes
  1  D local              127.0.0.1       100ms         none            no
yes
  2  D 10.0.0...          10.0.0.100      100ms         none            no
yes

[admin@Hautespot] ip telephony numbers> print
Flags: I - invalid, X - disabled, D - dynamic, R - registered
  #     DST-PATTERN             VOICE-PORT              PREFIX
  0     78                      linejack1
  1     3...                    vctx1
  2     33_                     voip1
  3     5..                     voip1
  4  XD 78                      local                   78
  5  XD 3_                      local                   3
  6   D 76                      10.0.0.100              76
  7   D 77                      10.0.0.100              77
  8   D 1_                      10.0.0.100              1
```

Here we can see how aliases and prefixes are added to numbers table. Entries 0..3 are static. Entries 4 and 5 are added by registering the local endpoint to the local gatekeeper. Entries 6..8 are added by registering endpoint (with IP address 10.0.0.100) to the local gatekeeper.

For prefixes, '_' is added at the end of dst-pattern to allow any additional digits to be added at the end.

Local endpoint is registered to the local gatekeeper too. So, local aliases and prefixes are added as dynamic numbers too. Only, as they are local and corresponding number entries already exist in the number table, then these dynamically added entries are disabled by default.

If any registered telephone number will conflict with some existing telephone numbers entry, it will be added as disabled and dynamic.

If in gatekeeper's numbers table there already exists exactly the same dst-pattern as some other endpoint is trying to register, this gatekeeper registration for that endpoint will fail.

Troubleshooting

Description

- **The IP Telephony does not work after upgrading from 2.5.x version** - You need to completely reinstall the router using any installation procedure.

You may keep the configuration using either the installation program option or the backup file.

- **The IP Telephony gateway does not detect the drop of the line when connected to some PBXs** - Different regional setting should be used to match the parameters of the PBX. For example, try using **uk** for Meridian PBX.
- **The IP Telephone does not call the gateway, but gives busy signal** - Enable the logging of IP telephony events under **/system logging facility**. Use the monitoring function for voice ports to debug your setup while making calls.
- **The IP telephony is working without NAT, but sound goes only in one direction** - Disable H323 service port in firewall: **/ip firewall service-port set h323 disabled=yes**
- **The IP Telephony does not work through NAT** - Enable H323 service port in firewall: **/ip firewall service-port set h323 disabled=no**

A simple example

Description

The following describes examples of some useful IP telephony applications using Hautespot RouterOS.

Let us consider the following example of IP telephony gateway, one Hautespot IP telephone, and one Welltech LAN Phone 101 setup:

Setting up the Hautespot IP Telephone

If you pick up the handset, a dialtone should be heard.

The basic telephony configuration should be as follows:

- Add a voip voice port to the **/ip telephony voice-port voip** for each of the devices you want to call, or want to receive calls from, i.e., (the IP telephony gateway 10.1.1.12 and the Welltech IP telephone 10.5.8.2):

```
[admin@Joe] ip telephony voice-port voip> add name=gw remote-address=10.1.1.12
[admin@Joe] ip telephony voice-port voip> add name=rob remote-address=10.5.8.2
[admin@Joe] ip telephony voice-port voip> print
Flags: X - disabled, D - dynamic, R - registered
  #    NAME      AUTODIAL REMOTE-ADDRESS  JITTER-BUFFER PREFERED-CODEC SIL FAS
```

```
  0    gw                  10.1.1.12       100ms         none
no  yes
  1    rob                 10.5.8.2        100ms         none
no  yes
[admin@Joe] ip telephony voice-port voip>
```

You should have three vioce ports now:

```
[admin@Joe] ip telephony voice-port> print
Flags: X - disabled
  #   NAME                          TYPE           AUTODIAL
  0   linejack1                     linejack
  1   gw                            voip
  2   rob                           voip
[admin@Joe] ip telephony voice-port>
```

- Add at least one unique number to the **/ip telephony numbers** for each voice port. This number will be used to call that port:

```
[admin@Joe] ip telephony numbers> add dst-pattern=31 voice-port=rob
[admin@Joe] ip telephony numbers> add dst-pattern=33 voice-
port=linejack1
[admin@Joe] ip telephony numbers> add dst-pattern=1. voice-port=gw
prefix=1
[admin@Joe] ip telephony numbers> print
Flags: I - invalid, X - disabled, D - dynamic, R - registered
  #     DST-PATTERN             VOICE-PORT              PREFIX
  0     31                          rob                 31
  1     33                          linejack1
  2     1.                          gw                  1
[admin@Joe] ip telephony numbers>
```

Here, the **dst-pattern=31** is to call the Welltech IP Telephone, if the number 31 is dialed on the dialpad. The **dst-pattern=33** is to ring the local telephone, if a call for number 33 is received over the network. Anything starting with digit '1' would be sent over to the IP Telephony gateway.

Making calls from the IP telephone 10.0.0.224:

- To call the IP telephone 10.5.8.2, it is enough to lift the handset and dial the number 31

- To call the PBX extension 13, it is enough to lift the handset and dial the number 13

  After establishing the connection with 13, the voice port monitor shows:

```
[admin@Joe] ip telephony voice-port linejack> monitor linejack
               status: connection
                 port: phone
            direction: port-to-ip
          line-status: unplugged
         phone-number: 13
    remote-party-name: PBX_Line [10.1.1.12]
                codec: G.723.1-6.3k/hw
             duration: 16s

[admin@Joe] ip telephony voice-port linejack>
```

Setting up the IP Telephony Gateway

The IP telephony gateway [voip_gw] requires the following configuration:

- Set the regional setting to match our PBX. The **Hautespot** region will be used in thisn example:

```
[admin@voip_gw] ip telephony voice-port linejack> set linejack1 region=Hautespot
[admin@voip_gw] ip telephony voice-port linejack> print
Flags: X - disabled
  0   name="linejack1" autodial="" region=Hautespot playback-volume=0
      record-volume=0 ring-cadence="++-++--- ++-++---" agc-on-playback=no
      agc-on-record=no aec=yes aec-tail-length=short aec-nlp-threshold=low
      aec-attenuation-scaling=4 aec-attenuation-boost=0 software-aec=no
      detect-cpt=yes

[admin@voip_gw] ip telephony voice-port linejack>
```

- Add a voip voice port to the **/ip telephony voice-port voip** for each of the devices you want to call, or want to receive calls from, i.e., (the IP telephone 10.0.0.224 and the Welltech IP telephone 10.5.8.2):

```
[admin@voip_gw] ip telephony voice-port voip> add name=joe \
\... remote-address=10.0.0.224
[admin@voip_gw] ip telephony voice-port voip> add name=rob \
\... remote-address=10.5.8.2 prefered-codec=G.723.1-6.3k/hw
```

```
[admin@voip_gw] ip telephony voice-port voip> print
Flags: X - disabled, D - dynamic, R - registered
  #    NAME      AUTODIAL REMOTE-ADDRESS  JITTER-BUFFER PREFERED-CODEC
SIL FAS
  0    joe               10.0.0.224      100ms         none
no yes
  1    rob               10.5.8.2        100ms         G.723.1-6.3k/hw
no yes
[admin@voip_gw] ip telephony voice-port voip>
```

- Add number records to the **/ip telephony numbers**, so you are able to make calls:

```
[admin@voip_gw] ip telephony numbers> add dst-pattern=31 voice-port=rob
prefix=31
[admin@voip_gw] ip telephony numbers> add dst-pattern=33 voice-port=joe
prefix=33
[admin@voip_gw] ip telephony numbers> add dst-pattern=1. voice-
port=linejack1 \
\... prefix=1
[admin@voip_gw] ip telephony numbers> print
Flags: I - invalid, X - disabled, D - dynamic, R - registered
  #     DST-PATTERN             VOICE-PORT              PREFIX
  0     31                          rob                 31
  1     33                          joe                 33
  2     1.                          linejack1           1
[admin@voip_gw] ip telephony numbers>
```

Making calls through the IP telephony gateway:

- To dial the IP telephone 10.0.0.224 from the office PBX line, the extension number 19 should be dialed, and, after the dial tone has been received, the number 33 should be entered. Thus, the telephone [Joe] is ringed.

  After establishing the voice connection with '33' (the call has been answered), the voice port monitor shows:

```
[admin@voip_gw] ip telephony voice-port linejack> monitor linejack1
               status: connection
                 port: line
            direction: port-to-ip
          line-status: plugged
         phone-number: 33
    remote-party-name: linejack1 [10.0.0.224]
```

```
                codec: G.723.1-6.3k/hw
             duration: 1m46s

[admin@voip_gw] ip telephony voice-port linejack>
```

- To dial the IP telephone 10.5.8.2 from the office PBX line, the extension number 19 should be dialed, and, after the dial tone has been received, the number 31 should be entered.

Setting up the Welltech IP Telephone

Please follow the documentation from www.welltech.com.tw on how to set up the Welltech LAN Phone 101. Here we give just brief recommendations:

1. We recommend to upgrade the Welltech LAN Phone 101 with the latest application software. Telnet to the phone and check what you have, for example:

```
usr/config$ rom -print

Download Method  :  TFTP
 Server Address  :  10.5.8.1

   Hardware Ver. :  4.0
        Boot Rom  :  nblp-boot.102a
Application Rom  :  wtlp.108h
        DSP App  :  48302ce3.127
     DSP Kernel  :  48302ck.127
  DSP Test Code  :  483cbit.bin
  Ringback Tone  :  wg-ringbacktone.100
      Hold Tone  :  wg-holdtone10s.100
  Ringing Tone1  :  ringlow.bin
  Ringing Tone2  :  ringmid.bin
  Ringing Tone3  :  ringhi.bin

usr/config$
```

20. Check if you have the codecs arranged in the desired order:

```
usr/config$ voice -print
Voice codec setting relate information
    Sending packet size  :
            G.723.1      : 30 ms
            G.711A       : 20 ms
            G.711U       : 20 ms
            G.729A       : 20 ms
            G.729        : 20 ms
    Priority order codec :
            g7231 g711a g711u g729a g729
    Volume levels        :
```

```
            voice volume : 54
            input gain   : 26
             dtmf volume : 23
Silence suppression & CNG:
            G.723.1      : Off
    Echo canceller       : On
 JitterBuffer Min Delay  : 90
 JitterBuffer Max Delay  : 150
usr/config$
```

41. Make sure you have set the H.323 operation mode to phone to phone (P2P), not gatekeeper (GK):

```
usr/config$ h323 -print
H.323 stack relate information
    RAS mode               : Non-GK mode
    Registered e164        : 31
    Registered H323 ID     : Rob
    RTP port               : 16384
    H.245 port             : 16640
    Allocated port range   :
              start port   : 1024
              end port     : 65535
    Response timeOut       : 5
    Connect  timeOut       : 5000
usr/config$
```

55. Add the gateway's address to the phonebook:

```
usr/config$ pbook -add name gw ip 10.1.1.12
usr/config$
This may take a few seconds, please wait....

Commit to flash memory ok!

usr/config$ pbook -print
index   Name                 IP                    E164
=========================================================================
1       gw                   10.1.1.12
-------------------------------------------------------------------------
usr/config$
```

Making calls from the IP telephone 10.5.8.2:

- Just lift the handset and dial '11', or '13' fo the PBX extensions.

- Dial '33' for [Joe]. The call request will be sent to the gateway 10.1.1.12, where it will be forwarded to [Joe]. If you want to call [Joe] directly, add a phonebook record for it:

```
usr/config$ pbook -add name Joe ip 10.0.0.224 e164 33
```

Use the telephony logging feature on the gateway to debug your setup.

Setting up Hautespot Router and CISCO Router

Let's try a different example.

Here are some hints on how to get working configuration for telephony calls between CISCO and Hautespot router.

Configuration on the Hautespot side

- G.729a codec MUST be disabled (otherwise connections are not possible at all!!!)

```
/ip telephony codec disable G.729A-8k/sw
```

- G.711-ALaw codec should not be used (in some cases there is no sound)

```
/ip telephony codec disable "G.711-ALaw-64k/sw G.711-ALaw-64k/hw"
```

- Fast start has to be used (otherwise no ring-back tone and problems with codec negotiation)

```
/ip telephony voice-port set cisco fast-start=yes
```

- Telephone number we want to call to must be sent to Cisco, for example

```
/ip telephony numbers add destination-pattern=101 voice-port=cisco
prefix=101
```

- Telephone number, cisco will call us, must be assigned to some voice port, for example,

```
/ip telephony numbers add destination-pattern=098 voice-port=linejack
```

Configuration on the CISCO side:

- IP routing has to be enabled

```
ip routing
```

- Default values for fast start can be used:

```
voice service pots
    default h323 call start
    exit
voice service voip
    default h323 call start
    exit
```

- Enable opening of RTP streams:

```
voice rtp send-recv
```

- Assign some E.164 number for local telephone, for example, 101 to port 0/0

```
dial-peer voice 1 pots
    destination-pattern 101
    port 0/0
    exit
```

- create preferred codec listing:

```
voice class codec codec_class_number
    codec preference 1 g711ulaw
    codec preference 2 g723r63
```

```
    exit
```

NOTE: g723r53 codec can be used, too

- Tell, that some foreign E.164 telephone number can be reached by calling to some IP address, for example, 098 by calling to 10.0.0.98

```
 dial-peer voice 11 voip
     destination-pattern 098
     session target ipv4:10.0.0.98
     voice-class codec codec_class_number
     exit
```

NOTE: instead of codec class, one specified codec could be specified:

```
codec g711ulaw
```

For reference, following is an exported CISCO configuration, that works:

```
!
version 12.1
no service single-slot-reload-enable
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname Router
!
logging rate-limit console 10 except errors
enable secret 5 $1$bTMC$nDGl9/n/pc3OMbtWxADMg1
enable password 123
!
memory-size iomem 25
ip subnet-zero
no ip finger
!
call rsvp-sync
voice rtp send-recv
!
voice class codec 1
 codec preference 1 g711ulaw
 codec preference 2 g723r63
!
interface FastEthernet0
```

```
 ip address 10.0.0.101 255.255.255.0
 no ip mroute-cache
 speed auto
 half-duplex
!
ip classless
ip route 0.0.0.0 0.0.0.0 10.0.0.1
no ip http server
!
dialer-list 1 protocol ip permit
dialer-list 1 protocol ipx permit
!
voice-port 0/0
!
voice-port 0/1
!
voice-port 2/0
!
voice-port 2/1
!
dial-peer voice 1 pots
 destination-pattern 101
 port 0/0
!
dial-peer voice 97 voip
 destination-pattern 097
 session target ipv4:10.0.0.97
 codec g711ulaw
!
dial-peer voice 98 voip
 destination-pattern 098
 voice-class codec 1
 session target ipv4:10.0.0.98
!
!
line con 0
 transport input none
line aux 0
line vty 0 4
 password 123
 login
!
end
```

Setting up PBX to PBX Connection over an IP Network

To interconnect two telephone switchboards (PBX) over an IP network, two IP telephony gateways should be configured. The setup is shown in the following diagram:

We want to be able to use make calls from local telephones of one PBX to local telephones or external lines of the other PBX.

Assume that:

- The IP telephony gateway #1 has IP address 10.0.0.182, and the name of the Voicetronix first line is 'vctx1'.
- The IP telephony gateway #2 has IP address 10.0.0.183, and the name of the Voicetronix first line is 'vctx1'.

The IP telephony configuration should be as follows:

- IP telephony gateway #1 should have:

```
/ip telephony voice-port voip
add name=gw2 remote-address=10.0.0.183
/ip telephony numbers
add dst-pattern=1.. voice-port=gw2 prefix=2
add dst-pattern=2.. voice-port=vctx1 prefix=1
```

- IP telephony gateway #2 should have

```
/ip telephony voice-port voip
add name=gw1 remote-address=10.0.0.182
/ip telephony numbers
add dst-pattern=2.. voice-port=vctx1 prefix=1
add dst-pattern=1.. voice-port=gw1 prefix=2
```

The system works as follows:

To dial from the main office PBX#1 any extension of the remote office PBX#2, the extension with the connected gateway at PBX#1 should be dialed first. Then, after the dial tone of the gateway#1 is received, the remote extension number should be dialed.

To dial from the main office PBX#2 any extension of the remote office PBX#1, the actions are the same as in first situation.

HauteRouterOS IPIP Tunnel Interfaces

Document revision: 1.1 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

The IPIP tunneling implementation on the HauteRouterOS is RFC 2003 compliant. IPIP tunnel is a simple protocol that encapsulates IP packets in IP to make a tunnel between two routers. The IPIP tunnel interface appears as an interface under the interface list. Many routers, including Cisco and Linux based, support this protocol. This protocol makes multiple network schemes possible.

IP tunneling protocol adds the following possibilities to a network setups:

- to tunnel Intranets over the Internet
- to use it instead of source routing

Quick Setup Guide

To make an IPIP tunnel between 2 Hautespot routers with IP addresses **10.5.8.104** and **10.1.0.172**, using IPIP tunnel addresses 10.0.0.1 and 10.0.0.2, follow the next steps.

- Configuration on router with IP address **10.5.8.104**:
  1. Add an IPIP interface (by default, its name will be **ipip1**):

```
2. [admin@10.5.8.104] interface ipip> add local-address=10.5.8.104 \
    remote-address=10.1.0.172 disabled=no
```

  3. Add an IP address to created **ipip1** interface:

```
[admin@10.5.8.104] ip address> add address=10.0.0.1/24
interface=ipip1
```

- Configuration on router with IP address **10.1.0.172**:
  1. Add an IPIP interface (by default, its name will be **ipip1**):

```
2. [admin@10.1.0.172] interface ipip> add local-address=10.1.0.172 \
    remote-address=10.5.8.104 disabled=no
```

3. Add an IP address to created **ipip1** interface:

```
[admin@10.1.0.172] ip address> add address=10.0.0.2/24
interface=ipip1
```

Specifications

Packages required: ***system***
License required: *Level1 (limited to 1 tunnel)* , *Level3 (200 tunnels)* , *Level5 (unlimited)*
Submenu level: ***/interface ipip***
Standards and Technologies: IPIP (RFC 2003)
Hardware usage: *Not significant*

Additional Resources

- http://www.ietf.org/rfc/rfc1853.txt?number=1853
- http://www.ietf.org/rfc/rfc2003.txt?number=2003
- http://www.ietf.org/rfc/rfc1241.txt?number=1241

IPIP Setup

Submenu level: ***/interface ipip***

Description

An IPIP interface should be configured on two routers that have the possibility for an IP level connection and are RFC 2003 compliant. The IPIP tunnel may run over any connection that transports IP. Each IPIP tunnel interface can connect with one remote router that has a corresponding interface configured. An unlimited number of IPIP tunnels may be added to the router. For more details on IPIP tunnels, see RFC 2003.

Property Description

name (*name*; default: **ipipN**) - interface name for reference
mtu (*integer*; default: **1480**) - Maximum Transmission Unit. Should be set to 1480 bytes to avoid fragmentation of packets. May be set to 1500 bytes if **mtu** path discovery is not working properly on links

local-address (*IP address*) - local address on router which sends IPIP traffic to the remote host
remote-address (*IP address*) - the IP address of the remote host of the IPIP tunnel - may be any RFC 2003 compliant router

Notes

Use **/ip address add** command to assign an **IP address** to the IPIP interface.

There is no authentication or 'state' for this interface. The bandwidth usage of the interface may be monitored with the **monitor** feature from the **interface** menu.

HauteRouterOS IPIP implementation has been tested with Cisco 1005. The sample of the Cisco 1005 configuration is given below:

```
interface Tunnel0
 ip address 10.3.0.1 255.255.255.0
 tunnel source 10.0.0.171
 tunnel destination 10.0.0.204
 tunnel mode ipip
```

Application Examples

Description

Suppose we want to add an IPIP tunnel between routers **R1** and **R2**:

At first, we need to configure IPIP interfaces and then add **IP addresses** to them.

The configuration for router **R1** is as follows:

```
[admin@Hautespot] interface ipip> add
local-address: 10.0.0.1
remote-address: 22.63.11.6
[admin@Hautespot] interface ipip> print
Flags: X - disabled, R - running
  #    NAME                                MTU   LOCAL-ADDRESS   REMOTE-ADDRESS
  0 X  ipip1                               1480  10.0.0.1        22.63.11.6

[admin@Hautespot] interface ipip> en 0
[admin@Hautespot] interface ipip> /ip address add address 1.1.1.1/24
interface=ipip1
```

The configuration of the **R2** is shown below:

```
[admin@Hautespot] interface ipip> add local-address=22.63.11.6 remote-
address=10.
0.0.1
[admin@Hautespot] interface ipip> print
Flags: X - disabled, R - running
  #    NAME                                MTU   LOCAL-ADDRESS    REMOTE-ADDRESS
  0 X  ipip1                               1480  22.63.11.6       10.0.0.1

[admin@Hautespot] interface ipip> enable 0
[admin@Hautespot] interface ipip> /ip address add address 1.1.1.2/24
interface=ipip1
```

Now both routers can ping each other:

```
[admin@Hautespot] interface ipip> /ping 1.1.1.2
1.1.1.2 64 byte ping: ttl=64 time=24 ms
1.1.1.2 64 byte ping: ttl=64 time=19 ms
1.1.1.2 64 byte ping: ttl=64 time=20 ms
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 19/21.0/24 ms
[admin@Hautespot] interface ipip>
```

HauteRouterOS L2TP Interface

Document revision: 1.1 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

L2TP (Layer 2 Tunnel Protocol) supports encrypted tunnels over IP. The HauteRouterOS implementation includes support for both L2TP client and server.

General applications of L2TP tunnels include:

- secure router-to-router tunnels over the Internet
- linking (bridging) local Intranets or LANs (in cooperation with EoIP)
- extending PPP user connections to a remote location (for example, to separate authentication and Internet access points for ISP)
- accessing an Intranet/LAN of a company for remote (mobile) clients (employees)

Each L2TP connection is composed of a server and a client. The HauteRouterOS may function as a server or client or, for various configurations, it may be the server for some connections and client for other connections.

Quick Setup Guide

To make a L2TP tunnel between 2 Hautespot routers with IP addresses **10.5.8.104** (L2TP server) and **10.1.0.172** (L2TP client), follow the next steps.

- Configuration on L2TP server router:
  1. Add a L2TP user:

```
2. [admin@L2TP-Server] ppp secret> add name=james password=pass \
```

```
\... local-address=10.0.0.1 remote-address=10.0.0.2
```

  3. Enable the L2TP server

```
[admin@L2TP-Server] interface l2tp-server server> set enabled=yes
```

- Configuration on L2TP client router:
  1. Add a L2TP client:

```
2. [admin@L2TP-Client] interface l2tp-client> add user=james
   password=pass \
```

```
\... connect-to=10.5.8.104
```

Specifications

Packages required: ***ppp***
License required: *Level1 (limited to 1 tunnel)* , *Level3 (limited to 200 tunnels)* , *Level5*
Submenu level: ***/interface l2tp-server***, ***/interface l2tp-client***
Standards and Technologies: L2TP (RFC 2661)
Hardware usage: *Not significant*

Description

L2TP is a secure tunnel protocol for transporting IP traffic using PPP. L2TP encapsulates PPP in virtual lines that run over IP, Frame Relay and other protocols (that are not currently supported by HauteRouterOS). L2TP incorporates PPP and MPPE (Microsoft Point to Point Encryption) to make encrypted links. The purpose of this protocol is to allow the Layer 2 and PPP endpoints to reside on different devices interconnected by a packet-switched network. With L2TP, a user has a Layer 2 connection to an access concentrator - **LAC** (e.g., modem bank, ADSL DSLAM, etc.),

and the concentrator then tunnels individual PPP frames to the Network Access Server - **NAS**. This allows the actual processing of PPP packets to be divorced from the termination of the Layer 2 circuit. From the user's perspective, there is no functional difference between having the L2 circuit terminate in a NAS directly or using L2TP.

It may also be useful to use L2TP just as any other tunneling protocol with or without encryption. The L2TP standard says that the most secure way to encrypt data is using L2TP over IPsec (**Note** that it is default mode for Microsoft L2TP client) as all L2TP control and data packets for a particular tunnel appear as homogeneous UDP/IP data packets to the IPsec system.

L2TP includes PPP authentication and accounting for each L2TP connection. Full authentication and accounting of each connection may be done through a RADIUS client or locally.

MPPE 40bit RC4 and MPPE 128bit RC4 encryption are supported.

L2TP traffic uses UDP protocol for both control and data packets. UDP port 1701 is used only for link establishment, further traffic is using any available UDP port (which may or may not be 1701). This means that L2TP can be used with most firewalls and routers (even with NAT) by enabling UDP traffic to be routed through the firewall or router.

L2TP Client Setup

Submenu level: ***/interface l2tp-client***

Property Description

name (*name*; default: **l2tp-outN**) - interface name for reference
mtu (*integer*; default: **1460**) - Maximum Transmission Unit. The optimal value is the MTU of the interface the tunnel is working over decreased by 40 (so, for 1500-byte Ethernet link, set the MTU to 1460 to avoid fragmentation of packets)
mru (*integer*; default: **1460**) - Maximum Receive Unit. The optimal value is the MRU of the interface the tunnel is working over decreased by 40 (so, for 1500-byte Ethernet link, set the MRU to 1460 to avoid fragmentation of packets)
connect-to (*IP address*) - The IP address of the L2TP server to connect to
user (*text*) - user name to use when logging on to the remote server
password (*text*; default: **""**) - user password to use when logging to the remote server
profile (*name*; default: **default**) - profile to use when connecting to the remote server
allow (*multiple choice:* mschap2, mschap1, chap, pap; default: **mschap2, mschap1, chap, pap**) - the protocol to allow the client to use for authentication
add-default-route (yes | no; default: **no**) - whether to use the server which this client is connected to as its default router (gateway)

Example

To set up L2TP client named **test2** using username **john** with password **john** to connect to the **10.1.1.12** L2TP server and use it as the default gateway:

```
[admin@Hautespot] interface l2tp-client> add name=test2 connect-to=10.1.1.12 \
\... user=john add-default-route=yes password=john
[admin@Hautespot] interface l2tp-client> print
Flags: X - disabled, R - running
  0 X  name="test2" mtu=1460 mru=1460 connect-to=10.1.1.12 user="john"
       password="john" profile=default add-default-route=yes


[admin@Hautespot] interface l2tp-client> enable 0
```

Monitoring L2TP Client

Command name: ***/interface l2tp-client monitor***

Property Description

status (*text*) - status of the client
Dialing - attempting to make a connection
Verifying password... - connection has been established to the server, password verification in progress
Connected - self-explanatory
Terminated - interface is not enabled or the other side will not establish a connection **uptime** (time) - connection time displayed in days, hours, minutes and seconds
encoding (*text*) - encryption and encoding (if asymmetric, separated with '/') being used in this connection

Example

Example of an established connection

```
[admin@Hautespot] interface l2tp-client> monitor test2
      status: "connected"
      uptime: 4m27s
    encoding: "MPPE128 stateless"
[admin@Hautespot] interface l2tp-client>
```

L2TP Server Setup

Submenu level: ***/interface l2tp-server server***

Description

The L2TP server creates a dynamic interface for each connected L2TP client. The L2TP connection count from clients depends on the license level you have. Level1 license allows 1 L2TP client, Level3 or Level4 licenses up to 200 clients, and Level5 or Level6 licenses do not have L2TP client limitations.

To create L2TP users, you should consult the PPP secret and PPP Profile manuals. It is also possible to use the Hautespot router as a RADIUS client to register the L2TP users, see the manual how to do it.

Property Description

enabled (yes | no; default: **no**) - defines whether L2TP server is enabled or not
mtu (*integer*; default: **1460**) - Maximum Transmission Unit. The optimal value is the MTU of the interface the tunnel is working over decreased by 40 (so, for 1500-byte Ethernet link, set the MTU to 1460 to avoid fragmentation of packets)
mru (*integer*; default: **1460**) - Maximum Receive Unit. The optimal value is the MRU of the interface the tunnel is working over decreased by 40 (so, for 1500-byte Ethernet link, set the MRU to 1460 to avoid fragmentation of packets)
authentication (*multiple choice:* pap | chap | mschap1 | mschap2; default: **mschap2**) - authentication algorithm
default-profile - default profile to use

Example

To enable L2TP server:

```
[admin@Hautespot] interface l2tp-server server> set enabled=yes
[admin@Hautespot] interface l2tp-server server> print
            enabled: yes
                mtu: 1460
                mru: 1460
     authentication: mschap2
    default-profile: default
[admin@Hautespot] interface l2tp-server server>
```

L2TP Server Users

Submenu level: ***/interface l2tp-server***

Description

There are two types of items in L2TP server configuration - static users and dynamic connections. A dynamic connection can be established if the user database or the **default-profile** has its **local-address** and **remote-address** set correctly. When static users are added, the default profile may be left with its default values and only PPP user (in **/ppp secret**) should be configured. **Note** that in both cases PPP users must be configured properly.

Property Description

name (*name*) - interface name
user (*text*) - the name of the user that is configured statically or added dynamically
mtu - shows client's MTU
client-address - shows the IP of the connected client
uptime - shows how long the client is connected
encoding (*text*) - encryption and encoding (if asymmetric, separated with '/') being used in this connection

Example

To add a static entry for **ex1** user:

```
[admin@Hautespot] interface l2tp-server> add user=ex1
[admin@Hautespot] interface l2tp-server> print
Flags: X - disabled, D - dynamic, R - running
  #     NAME                 USER         MTU   CLIENT-ADDRESS  UPTIME
ENC...
  0  DR <l2tp-ex>            ex           1460  10.0.0.202      6m32s    none
  1     l2tp-in1             ex1
[admin@Hautespot] interface l2tp-server>
```

In this example an already connected user **ex** is shown besides the one we just added.

L2TP Application Examples

Router-to-Router Secure Tunnel Example

There are two routers in this example:

- [HomeOffice]

  Interface LocalHomeOffice 10.150.2.254/24

  Interface ToInternet 192.168.80.1/24

- [RemoteOffice]

  Interface ToInternet 192.168.81.1/24

  Interface LocalRemoteOffice 10.150.1.254/24

Each router is connected to a different ISP. One router can access another router through the Internet.

On the L2TP server a user must be set up for the client:

```
[admin@HomeOffice] ppp secret> add name=ex service=l2tp password=lkjrht
local-address=10.0.103.1 remote-address=10.0.103.2
[admin@HomeOffice] ppp secret> print detail
Flags: X - disabled
  0   name="ex" service=l2tp caller-id="" password="lkjrht" profile=default
      local-address=10.0.103.1 remote-address=10.0.103.2 routes==""

[admin@HomeOffice] ppp secret>
```

Then the user should be added in the L2TP server list:

```
[admin@HomeOffice] interface l2tp-server> add user=ex
[admin@HomeOffice] interface l2tp-server> print
Flags: X - disabled, D - dynamic, R - running
  #     NAME                USER         MTU   CLIENT-ADDRESS  UPTIME
ENC...
  0     l2tp-in1            ex
[admin@HomeOffice] interface l2tp-server>
```

And finally, the server must be enabled:

```
[admin@HomeOffice] interface l2tp-server server> set enabled=yes
[admin@HomeOffice] interface l2tp-server server> print
            enabled: yes
                mtu: 1460
                mru: 1460
     authentication: mschap2
    default-profile: default
[admin@HomeOffice] interface l2tp-server server>
```

Add a L2TP client to the RemoteOffice router:

```
[admin@RemoteOffice] interface l2tp-client> add connect-to=192.168.80.1
user=ex \
\... password=lkjrht disabled=no
[admin@RemoteOffice] interface l2tp-client> print
Flags: X - disabled, R - running
  0  R name="l2tp-out1" mtu=1460 mru=1460 connect-to=192.168.80.1 user="ex"
       password="lkjrht" profile=default add-default-route=no


[admin@RemoteOffice] interface l2tp-client>
```

Thus, a L2TP tunnel is created between the routers. This tunnel is like an Ethernet point-to-point connection between the routers with IP addresses 10.0.103.1 and 10.0.103.2 at each router. It enables 'direct' communication between the routers over third party networks.

To route the local Intranets over the L2TP tunnel you need to add these routes:

```
[admin@HomeOffice] > ip route add dst-address 10.150.1.0/24 gateway 10.0.103.2
[admin@RemoteOffice] > ip route add dst-address 10.150.2.0/24 gateway
10.0.103.1
```

On the L2TP server it can alternatively be done using **routes** parameter of the user configuration:

```
[admin@HomeOffice] ppp secret> print detail
Flags: X - disabled
  0   name="ex" service=l2tp caller-id="" password="lkjrht" profile=default
      local-address=10.0.103.1 remote-address=10.0.103.2 routes==""

[admin@HomeOffice] ppp secret> set 0 routes="10.150.1.0/24 10.0.103.2 1"
[admin@HomeOffice] ppp secret> print detail
Flags: X - disabled
  0   name="ex" service=l2tp caller-id="" password="lkjrht" profile=default
      local-address=10.0.103.1 remote-address=10.0.103.2
      routes="10.150.1.0/24 10.0.103.2 1"

[admin@HomeOffice] ppp secret>
```

Test the L2TP tunnel connection:

```
[admin@RemoteOffice]> /ping 10.0.103.1
10.0.103.1 pong: ttl=255 time=3 ms
10.0.103.1 pong: ttl=255 time=3 ms
10.0.103.1 pong: ttl=255 time=3 ms
ping interrupted
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 3/3.0/3 ms
```

Test the connection through the L2TP tunnel to the LocalHomeOffice interface:

```
[admin@RemoteOffice]> /ping 10.150.2.254
10.150.2.254 pong: ttl=255 time=3 ms
10.150.2.254 pong: ttl=255 time=3 ms
10.150.2.254 pong: ttl=255 time=3 ms
ping interrupted
3 packets transmitted, 3 packets received, 0% packet loss
```

```
round-trip min/avg/max = 3/3.0/3 ms
```

To bridge a LAN over this secure tunnel, please see the example in the 'EoIP' section of the manual. To set the maximum speed for traffic over this tunnel, please consult the 'Queues' section.

Connecting a Remote Client via L2TP Tunnel

The following example shows how to connect a computer to a remote office network over L2TP encrypted tunnel giving that computer an IP address from the same network as the remote office has (without need of bridging over EoIP tunnels).

Please, consult the respective manual on how to set up a L2TP client with the software you are using.

The router in this example:

- [RemoteOffice]

  Interface ToInternet 192.168.81.1/24

  Interface Office 10.150.1.254/24

The client computer can access the router through the Internet.

On the L2TP server a user must be set up for the client:

```
[admin@RemoteOffice] ppp secret> add name=ex service=l2tp password=lkjrht
local-address=10.150.1.254 remote-address=10.150.1.2
[admin@RemoteOffice] ppp secret> print detail
Flags: X - disabled
  0   name="ex" service=l2tp caller-id="" password="lkjrht" profile=default
      local-address=10.150.1.254 remote-address=10.150.1.2 routes==""

[admin@RemoteOffice] ppp secret>
```

Then the user should be added in the L2TP server list:

```
[admin@RemoteOffice] interface l2tp-server> add name=FromLaptop user=ex
[admin@RemoteOffice] interface l2tp-server> print
Flags: X - disabled, D - dynamic, R - running
  #     NAME                USER         MTU   CLIENT-ADDRESS  UPTIME
ENC...
  0     FromLaptop          ex
```

```
[admin@RemoteOffice] interface l2tp-server>
```

And the server must be enabled:

```
[admin@RemoteOffice] interface l2tp-server server> set enabled=yes
[admin@RemoteOffice] interface l2tp-server server> print
            enabled: yes
                mtu: 1460
                mru: 1460
     authentication: mschap2
    default-profile: default
[admin@RemoteOffice] interface l2tp-server server>
```

Finally, the proxy APR must be enabled on the 'Office' interface:

```
[admin@RemoteOffice] interface ethernet> set Office arp=proxy-arp
[admin@RemoteOffice] interface ethernet> print
Flags: X - disabled, R - running
  #    NAME                MTU   MAC-ADDRESS       ARP
  0  R ToInternet          1500  00:30:4F:0B:7B:C1 enabled
  1  R Office              1500  00:30:4F:06:62:12 proxy-arp
[admin@RemoteOffice] interface ethernet>
```

L2TP Setup for Windows

Microsoft provides L2TP client support for Windows XP, 2000, NT4, ME and 98. Windows 2000 and XP include support in the Windows setup or automatically install L2TP. For 98, NT and ME, installation requires a download from Microsoft (L2TP/IPsec VPN Client).

For more information, see:

Microsoft L2TP/IPsec VPN Client Microsoft L2TP/IPsec VPN Client

On Windows 2000, L2TP setup without IPsec requires editing registry:

Disabling IPsec for the Windows 2000 Client

Disabling IPSEC Policy Used with L2TP

Troubleshooting

Description

- **I use firewall and I cannot establish L2TP connection**

Make sure UDP connections can pass through both directions between your sites.

- **My Windows L2TP/IPsec VPN Client fails to connect to L2TP server with "Error 789" or "Error 781"**

  The error messages 789 and 781 occur when IPsec is not configured properly on both ends. See the respective documentation on how to configure IPsec in the Microsoft L2TP/IPsec VPN Client and in the HauteRouterOS. If you do not want to use IPsec, it can be easily switched off on the client side. Note: if you are using Windows 2000, you need to edit system registry using regedt32.exe or regedit.exe. Add the following registry value to **HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Rasman\Parameters**:

```
Value Name: ProhibitIpSec
Data Type: REG_DWORD
Value: 1
```

You must restart the Windows 2000 for the changes to take effect

For more information on configuring Windows 2000, see:

- Configuring Cisco IOS and Windows 2000 Clients for L2TP Using Microsoft IAS
- Disabling IPSEC Policy Used with L2TP
- How to Configure a L2TP/IPsec Connection Using Pre-shared Key Authentication

HauteRouterOS Log Management

Document revision: 2.3 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

Various system events and status information can be logged. Logs can be saved in local routers file, displayed in console, sent to an email or to a remote server running a syslog daemon. HauteSpot Networks provides a shareware Windows Syslog daemon, which can be downloaded from www.hautespot.net

Specifications

Packages required: ***system***
License required: *Level1*
Submenu level: ***/system logging***, ***/log***
Standards and Technologies: Syslog
Hardware usage: *Not significant*

Description

Logs have different groups or topics. Logs from each topic can be configured to be discarded, logged locally or remotely. Locally log files can be stored in memory (default; logs are lost on reboot) or on hard drive (not enabled by default as is harmful for flash disks).

General Settings

Submenu level: ***/system logging***

Property Description

topics (info | critical | firewall | keepalive | packet | read | timer | write | ddns | hotspot | l2tp | ppp | route | update | account | debug | ike | manager | pppoe | script | warning | async | dhcp | info | notification | pptp | state | watchdog | bgp | error | ipsec | open | radius | system | web-proxy | calc | event | isdn | ospf | raw | telephony | wireless; default: **info**) - specifies log group or log message type
action (disk | echo | memory | remote; default: **memory**) - specifies one of the system actions or user specified action listed in **/system logging action**
prefix (*name*) - local log prefix

Example

To logg messages that are generated by firewall by saving them in local buffer

```
[admin@Hautespot] system logging> add topics=firewall action=memory
[admin@Hautespot] system logging> print
 Flags: X - disabled, I - invalid
 #   TOPICS                                             ACTION PREFIX
 0   info                                               memory
 1   error                                              memory
 2   warning                                            memory
 3   critical                                           echo
 4   firewall                                           memory
[admin@Hautespot] system logging>
```

Actions

Submenu level: ***/system logging action***

Property Description

disk-lines (*integer*; default: **100**) - Used when target is set to type disk. Specifies the number of records in log file
disk-stop-on-full (yes | no; default: **no**) - Used when target is set to type disk. Specifies whether to stop to save log messages on disk after the specified disk-lines number is reached
email-to (*name*) - Used when target is set to type email, sets email address logs are sent to
memory-lines (*integer*; default: **100**) - Used when target is set to type memory. Specifies the number of records in local buffer.
memory-stop-on-full (yes | no; default: **no**) - Used when target is set to type memory. Specifies whether to stop to save log messages in local buffer after the specified memory-lines number is reached
name (*name*) - name of an action
remember (yes | no; default: **yes**) - Used when target is set to type echo. Specifies whether to keep log messages, which have not yet been displayed in console
remote (*IP address*:*port* | *IP address*:*integer*: 0..65535; default: **0.0.0.0:514**) - Used when target is set to type remote. Remote log server's IP address and UDP port
target (disk | echo | email | memory | remote; default: **memory**) - Specifies how to treat logs
disk - logs are saved to hard drive
echo - logs are displayed in console
email - logs are sent by email
memory - logs are saved to local buffer. They can be viewed using the '/log print' command
remote - logs are sent to remote host

Notes

You cannot delete or rename default actions.

Example

To add a new action with name short, that will save logs in local buffer, if number of records in buffer are less than 50:

```
[admin@Hautespot] system logging action> add name=short \
\... target=memory memory-lines=50 memory-stop-on-full=yes
[admin@Hautespot] system logging action> print
 # FACILITY       LOCAL  REMOTE PREFIX         REMOTE-ADDRESS  REMOTE-PORT
ECHO
 Flags: * - default
 #   NAME                              TARGET REMOTE
 0 * memory                            memory
 1 * disk                              disk
 2 * echo                              echo
 3 * remote                            remote 0.0.0.0:514
 4   short                             memory
```

```
[admin@Hautespot] system logging action>
```

Log Messages

Submenu level: ***/log***

Description

Displays locally stored log messages

Property Description

message (*text*) - message text
time (*text*) - date and time of the event

Command Description

print - shows log messages
- **buffer** - prints log messages that were saved in specified local buffer
- **follow** - monitor system logs
- **without-paging** - prints logs without paging
- **file** - saves the log information on local ftp server with a specified file name

Example

To view the local logs:

```
[admin@Hautespot] > log print
 TIME                 MESSAGE
 dec/24/2003 08:20:36 log configuration changed by admin
 dec/24/2003 08:20:36 log configuration changed by admin
 dec/24/2003 08:20:36 log configuration changed by admin
 dec/24/2003 08:20:36 log configuration changed by admin
 dec/24/2003 08:20:36 log configuration changed by admin
 dec/24/2003 08:20:36 log configuration changed by admin
-- [Q quit|D dump]
```

To monitor the system log:

```
[admin@Hautespot] > log print follow
 TIME                 MESSAGE
 dec/24/2003 08:20:36 log configuration changed by admin
 dec/24/2003 08:24:34 log configuration changed by admin
 dec/24/2003 08:24:51 log configuration changed by admin
 dec/24/2003 08:25:59 log configuration changed by admin
 dec/24/2003 08:25:59 log configuration changed by admin
```

```
 dec/24/2003 08:30:05 log configuration changed by admin
 dec/24/2003 08:30:05 log configuration changed by admin
 dec/24/2003 08:35:56 system started
 dec/24/2003 08:35:57 isdn-out1: initializing...
 dec/24/2003 08:35:57 isdn-out1: dialing...
 dec/24/2003 08:35:58 Prism firmware loading: OK
 dec/24/2003 08:37:48 user admin logged in from 10.1.0.60 via telnet
-- Ctrl-C to quit. New entries will appear at bottom.
```

HauteRouterOS MAC Level Access (Telnet and HSConfig)

Document revision: 2.2 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

MAC telnet is used to provide access to a router that has no IP address set. It works just like IP telnet. MAC telnet is possible between two HauteRouterOS routers only.

Specifications

Packages required: ***system***
License required: *Level1*
Submenu level: ***/tool***, ***/tool mac-server***
Standards and Technologies: MAC Telnet
Hardware usage: *Not significant*

MAC Telnet Server

Submenu level: ***/tool mac-server***

Property Description

interface (name | all; default: **all**) - interface name to which the mac-server clients will connect
all - all interfaces

Notes

There is an interface list in this submenu level. If you add some interfaces to this list, you allow MAC telnet to that interface. Disabled (**disabled=yes**) item means that interface is not allowed to accept MAC telnet sessions on that interface.

Example

To enable MAC telnet server on **ether1** interface only:

```
[admin@HauteSpot] tool mac-server> print
Flags: X - disabled
 #   INTERFACE
 0   all
[admin@HauteSpot] tool mac-server> remove 0
[admin@HauteSpot] tool mac-server> add interface=ether1 disabled=no
[admin@HauteSpot] tool mac-server> print
Flags: X - disabled
 #   INTERFACE
 0   ether1
[admin@HauteSpot] tool mac-server>
```

MAC HSConfig Server

Submenu level: ***/tool mac-server mac-HSConfig***

Property Description

interface (name | all; default: **all**) - interface name to which it is alowed to connect with HSConfig using MAC-based protocol
all - all interfaces

Notes

There is an interface list in this submenu level. If you add some interfaces to this list, you allow MAC HSConfig to that interface. Disabled (**disabled=yes**) item means that interface is not allowed to accept MAC HSConfig sessions on that interface.

Example

To enable MAC HSConfig server on **ether1** interface only:

```
[admin@HauteSpot] tool mac-server mac-HSConfig> print
Flags: X - disabled
 #   INTERFACE
 0   all
[admin@HauteSpot] tool mac-server mac-HSConfig> remove 0
[admin@HauteSpot] tool mac-server mac-HSConfig> add interface=ether1
disabled=no
[admin@HauteSpot] tool mac-server mac-HSConfig> print
```

```
Flags: X - disabled
 #   INTERFACE
 0   ether1
[admin@HauteSpot] tool mac-server mac-HSConfig>
```

Monitoring Active Session List

Submenu level: ***/tool mac-server sessions***

Property Description

interface (*read-only: name*) - interface to which the client is connected to
src-address (*read-only: MAC address*) - client's MAC address
uptime (*read-only: time*) - how long the client is connected to the server

Example

To see active MAC Telnet sessions:

```
[admin@HauteSpot] tool mac-server sessions> print
 # INTERFACE SRC-ADDRESS       UPTIME
 0 wlan1     00:0B:6B:31:08:22 00:03:01
[admin@HauteSpot] tool mac-server sessions>
```

MAC Telnet Client

Command name: ***/tool mac-telnet [MAC-address]***

Example

```
[admin@HauteSpot] > /tool mac-telnet 00:02:6F:06:59:42
Login: admin
Password:
Trying 00:02:6F:06:59:42...
Connected to 00:02:6F:06:59:42

  HauteRouterOS 2.9 (c) 1999-2004           http://www.HauteSpot.com/

Terminal linux detected, using multiline input mode
[admin@HauteSpot] >
```

HauteRouterOS Mangle

Document revision: 3 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

The mangle facility allows to mark IP packets with special marks. These marks are used by various other router facilities to identify the packets. Additionally, the mangle facility is used to modify some fields in the IP header, like TOS (DSCP) and TTL fields.

Specifications

Packages required: ***system***
License required: *Level1*
Submenu level: ***/ip firewall mangle***
Standards and Technologies: IP
Hardware usage: *Increases with count of mangle rules*

Mangle

Submenu level: ***/ip firewall mangle***

Description

Mangle is a kind of 'marker' that marks packets for future processing with special marks. Many other facilities in RouterOS make use of these marks, e.g. queue trees and NAT. They identify a packet based on its mark and process it accordingly. The mangle marks exist only within the router, they are not transmitted across the network.

Property Description

action (accept | add-dst-to-address-list | add-src-to-address-list | change-mss | change-tos | change-ttl | jump | log | mark-connection | mark-packet | mark-routing | passthrough | return | strip-ipv4-options; default: **accept**) - action to undertake if the packet matches the rule
accept - accept the packet. No action, i.e., the packet is passed through and no more rules are applied to it
add-dst-to-address-list - add destination address of an IP packet to the address list specified by **address-list** parameter
add-src-to-address-list - add source address of an IP packet to the address list specified by **address-list** parameter
change-mss - change Maximum Segment Size field value of the packet to a value specified by the **new-mss** parameter
change-tos - change Type of Service field value of the packet to a value specified by the **new-tos**

parameter
change-ttl - change Time to Live field value of the packet to a value specified by the **new-ttl** parameter
jump - jump to the chain specified by the value of the **jump-target** parameter
log - each match with this action will add a message to the system log
mark-connection - place a mark specified by the **new-connection-mark** parameter on the entire connection that matches the rule
mark-packet - place a mark specified by the **new-packet-mark** parameter on a packet that matches the rule
mark-routing - place a mark specified by the **new-routing-mark** parameter on a packet. This kind of marks is used for policy routing purposes only
passthrough - ignore this rule go on to the next one
return - pass control back to the chain from where the jump took place
strip-ipv4-options - strip IPv4 option fields from the IP packet
address-list (*name*) - specify the name of the address list to collect IP addresses from rules having **action=add-dst-to-address-list** or **action=add-src-to-address-list** actions. These address lists could be later used for packet matching
address-list-timeout (*time*; default: **00:00:00**) - time interval after which the address will be removed from the address list specified by **address-list** parameter. Used in conjunction with **add-dst-to-address-list** or **add-src-to-address-list** actions
00:00:00 - leave the address in the address list forever
chain (forward | input | output | postrouting | prerouting) - specify the chain to put a particular rule into. As the different traffic is passed through different chains, always be careful in choosing the right chain for a new rule. If the input does not match the name of an already defined chain, a new chain will be created
comment (*text*) - free form textual comment for the rule. A comment can be used to refer the particular rule from scripts
connection-bytes (*integer-integer*) - match packets only if a given amount of bytes has been transfered through the particular connection
0 - means infinity, *exempli gratia*: **connection-bytes=2000000-0** means that the rule matches if more than 2MB has been transfered through the relevant connection
connection-limit (*integer*,*netmask*) - restrict connection limit per address or address block
connection-mark (*name*) - match packets marked via mangle facility with particular connection mark
connection-type (ftp | gre | h323 | irc | mms | pptp | quake3 | tftp) - match packets from related connections based on information from their connection tracking helpers. A relevant connection helper must be enabled under **/ip firewall service-port**
content (*text*) - the text packets should contain in order to match the rule
dst-address (*IP address*/*netmask* | *IP address-IP address*) - specify the address range an IP packet is destined to. Note that console converts entered **address/netmask** value to a valid network address, i.e.:**1.1.1.1/24** is converted to **1.1.1.0/24**
dst-address-list (*name*) - match destination address of a packet against user-defined address list
dst-address-type (unicast | local | broadcast | multicast) - match destination address type of the IP packet, one of the:
unicast - IP addresses used for one point to another point transmission. There is only one sender and one receiver in this case
local - match addresses assigned to router's interfaces
broadcast - the IP packet is sent from one point to all other points in the IP subnetwork

multicast - this type of IP addressing is responsible for transmission from one or more points to a set of other points

dst-limit (*integer*/*time*{0,1},*integer*,dst-address | dst-port | src-address{+},*time*{0,1}) - limit the packet per second (pps) rate on a per destination IP or per destination port base. As opposed to the **limit** match, every destination IP address / destination port has it's own limit. The options are as follows (in order of appearance):

Count - maximum average packet rate, measured in packets per second (pps), unless followed by **Time** option

Time - specifies the time interval over which the packet rate is measured

Burst - number of packets to match in a burst

Mode - the classifier(-s) for packet rate limiting

Expire - specifies interval after which recorded IP addresses / ports will be deleted

dst-port (*integer*: 0..65535-*integer*: 0..65535{*}) - destination port number or range

hotspot (*multiple choice:* from-client | auth | local-dst | http) - match packets received from clients against various Hot-Spot. All values can be negated

from-client - true, if a packet comes from HotSpot client

auth - true, if a packet comes from authenticted client

local-dst - true, if a packet has local destination IP address

hotspot - true, if it is a TCP packet from client and either the transparent proxy on port 80 is enabled or the client has a proxy address configured and this address is equal to the address:port pair of the IP packet

icmp-options (*integer*:*integer*) - match ICMP Type:Code fields

in-interface (*name*) - interface the packet has entered the router through

ipv4-options (any | loose-source-routing | no-record-route | no-router-alert | no-source-routing | no-timestamp | none | record-route | router-alert | strict-source-routing | timestamp) - match ipv4 header options

any - match packet with at least one of the ipv4 options

loose-source-routing - match packets with loose source routing option. This option is used to route the internet datagram based on information supplied by the source

no-record-route - match packets with no record route option. This option is used to route the internet datagram based on information supplied by the source

no-router-alert - match packets with no router alter option

no-source-routing - match packets with no source routing option

no-timestamp - match packets with no timestamp option

record-route - match packets with record route option

router-alert - match packets with router alter option

strict-source-routing - match packets with strict source routing option

timestamp - match packets with timestamp

jump-target (forward | input | output | postrouting | prerouting*name*) - name of the target chain to jump to, if the **action=jump** is used

limit (*integer*/*time*{0,1},*integer*) - restrict packet match rate to a given limit. Usefull to reduce the amount of log messages

Count - maximum average packet rate, measured in packets per second (pps), unless followed by **Time** option

Time - specify the time interval over which the packet rate is measured

Burst - number of packets to match in a burst

log-prefix (*text*) - all messages written to logs will contain the prefix specified herein. Used in conjunction with **action=log**

new-connection-mark (*name*) - specify the new value of the connection mark to be used in conjunction with **action=mark-connection**

new-mss (*integer*) - specify MSS value to be used in conjunction with **action=change-mss**

new-packet-mark (*name*) - specify the new value of the packet mark to be used in conjunction with **action=mark-packet**
new-routing-mark (*name*) - specify the new value of the routing mark used in conjunction with **action=mark-routing**
new-tos (max-reliability | max-throughput | min-cost | min-delay | normal *integer*) - specify TOS value to be used in conjunction with **action=change-tos**
max-reliability - maximize reliability (ToS=4)
max-throughput - maximize throughput (ToS=8)
min-cost - minimize monetary cost (ToS=2)
min-delay - minimize delay (ToS=16)
normal - normal service (ToS=0)
new-ttl (decrement | increment | set:*integer*) - specify the new TTL field value used in conjunction with **action=change-ttl**
decrement - the value of the TTL field will be decremented for *value*
increment - the value of the TTL field will be incremented for *value*
set: - the value of the TTL field will be set to *value*
nth (*integer*,*integer*: 0..15,*integer*{0,1}) - match a particular Nth packet received by the rule. One of 16 available counters can be used to count packets
Every - match every **Every+1**th packet. For example, if **Every=1** then the rule matches every 2nd packet
Counter - specifies which counter to use. A counter increments each time the rule containing **nth** match matches
Packet - match on the given packet number. The value by obvious reasons must be between **0** and **Every**. If this option is used for a given counter, then there must be at least **Every+1** rules with this option, covering all values between **0** and **Every** inclusively.
out-interface (*name*) - match the interface name a packet left the router through
p2p (all-p2p | bit-torrent | direct-connect | edonkey | fasttrack | gnutella | soulseek | warez | winmx) - match packets belonging to connections of the above P2P protocols
packet-mark (*name*) - match the packets marked in mangle with specific packet mark
packet-size (*integer*: 0..65535-*integer*: 0..65535{0,1}) - matches packet of the specified size or size range in bytes
Min - specifies lower boundary of the size range or a standalone value
Max - specifies upper boundary of the size range
passthrough (yes | no; default: **yes**) - whether to let the packet to pass further (like action **passthrough**) after marking it with a given mark (property only valid if action is mark packet, connection or routing mark)
phys-in-interface (*name*) - matches the bridge port physical input device added to a bridge device. It is only useful if the packet has arrived through the bridge
protocol (ddp | egp | encap | ggp | gre | hmp | icmp | idrp-cmtp | igmp | ipencap | ipip | ipsec-ah | ipsec-esp | iso-tp4 | ospf | pup | rdp | rspf | st | tcp | udp | vmtp | xns-idp | xtp | *integer*) - matches particular IP protocol specified by protocol name or number. You should specify this setting if you want to specify ports
psd (*integer*,*time*,*integer*,*integer*) - attempts to detect TCP and UDP scans. It is advised to assign lower weight to ports with high numbers to reduce the frequency of false positives, such as from passive mode FTP transfers
WeightThreshold - total weight of the latest TCP/UDP packets with different destination ports coming from the same host to be treated as port scan sequence
DelayThreshold - delay for the packets with different destination ports coming from the same host to be treated as possible port scan subsequence
LowPortWeight - weight of the packets with privileged (<=1024) destination port
HighPortWeight - weight of the packet with non-priviliged destination port
random (*integer*: 1..99) - matches packets randomly with given propability

routing-mark (*name*) - matches packets marked with the specified routing mark
src-address (*IP address/netmask* | *IP address-IP address*) - specifies the address range an IP packet is originated from. Note that console converts entered **address/netmask** value to a valid network address, i.e.:**1.1.1.1/24** is converted to **1.1.1.0/24**
src-address-list (*name*) - matches source address of a packet against user-defined address list
src-address-type (unicast | local | broadcast | multicast) - matches source address type of the IP packet, one of the:
unicast - IP addresses used for one point to another point transmission. There is only one sender and one receiver in this case
local - matches addresses assigned to router's interfaces
broadcast - the IP packet is sent from one point to all other points in the IP subnetwork
multicast - this type of IP addressing is responsible for transmission from one or more points to a set of other points
src-mac-address (*MAC address*) - source MAC address
src-port (*integer*: 0..65535-*integer*: 0..65535{*}) - source port number or range
tcp-flags (*multiple choice:* ack | cwr | ece | fin | psh | rst | syn | urg) - tcp flags to match
ack - acknowledging data
cwr - congestion window reduced
ece - ECN-echo flag (explicit congestion notification)
fin - close connection
psh - push function
rst - drop connection
syn - new connection
urg - urgent data
tcp-mss (*integer*: 0..65535) - matches TCP MSS value of an IP packet
time (*time-time*,sat | fri | thu | wed | tue | mon | sun{+}) - allows to create filter based on the packets' arrival time and date or, for locally generated packets, departure time and date
tos (max-reliability | max-throughput | min-cost | min-delay | normal) - specifies a match for the value of Type of Service (ToS) field of an IP header
max-reliability - maximize reliability (ToS=4)
max-throughput - maximize throughput (ToS=8)
min-cost - minimize monetary cost (ToS=2)
min-delay - minimize delay (ToS=16)
normal - normal service (ToS=0)

Notes

Instead of making two rules if you want to mark a packet, connection or routing-mark and finish mangle table processing on that event (in other words, mark and simultaneously accept the packet), you may disable the set by default **passthrough** property of the marking rule.

Usually routing-mark is not used for P2P, since P2P traffic always is routed over a default getaway.

Application Examples

Description

The following section discusses some examples of using the mangle facility.

Peer-to-Peer Traffic Marking

To ensure the quality of service for network connection, interactive traffic types such as VoIP and HTTP should be prioritized over non-interactive, such as peer-to-peer network traffic. RouterOS QOS implementation uses mangle to mark different types of traffic first, and then place them into queues with different limits.

The following example enforces the P2P traffic will get no more than 1Mbps of the total link capacity when the link is heavily used by other traffic otherwice expanding to the full link capacity:

```
[admin@Hautespot] > /ip firewall mangle add chain=forward \
\... p2p=all-p2p action=mark-connection new-connection-mark=p2p_conn
[admin@Hautespot] > /ip firewall mangle add chain=forward \
\... connection-mark=p2p_conn action=mark-packet new-packet-mark=p2p
[admin@Hautespot] > /ip firewall mangle add chain=forward \
\... connection-mark=!p2p_conn action=mark-packet new-packet-mark=other
[admin@Hautespot] > /ip firewall mangle print
Flags: X - disabled, I - invalid, D - dynamic
 0   chain=forward p2p=all-p2p action=mark-connection new-connection-
mark=p2p_conn

 1   chain=forward connection-mark=p2p_conn action=mark-packet new-packet-
mark=p2p

 2   chain=forward packet-mark=!p2p_conn action=mark-packet new-packet-
mark=other
[admin@Hautespot] >
[admin@Hautespot] > /queue tree add parent=Public packet-mark=p2p limit-
at=1000000 \
\... max-limit=100000000 priority=8
[admin@Hautespot] > /queue tree add parent=Local packet-mark=p2p limit-
at=1000000 \
\... max-limit=100000000 priority=8
[admin@Hautespot] > /queue tree add parent=Public packet-mark=other  limit-
at=1000000 \
\... max-limit=100000000 priority=1
[admin@Hautespot] > /queue tree add parent=Local packet-mark=other  limit-
at=1000000 \
\... max-limit=100000000 priority=1
```

Mark by MAC address

To mark traffic from a known MAC address which goes to the router or through it, do the following:

```
[admin@Hautespot] > / ip firewall mangle add chain=prerouting \
```

```
\... src-mac-address=00:01:29:60:36:E7 action=mark-connection new-connection-
mark=known_mac_conn
[admin@Hautespot] > / ip firewall mangle add chain=prerouting \
\... connection-mark=known_mac_conn action=mark-packet new-packet-
mark=known_mac
```

Change MSS

It is a well known fact that VPN links have smaller packet size due to incapsulation overhead. A large packet with MSS that exceeds the MSS of the VPN link should be fragmented prior to sending it via that kind of connection. However, if the packet has DF flag set, it cannot be fragmented and should be discarded. On links that have broken path MTU discovery (PMTUD) it may lead to a number of problems, including problems with FTP and HTTP data transfer and e-mail services.

In case of link with broken PMTUD, a decrease of the MSS of the packets coming through the VPN link solves the problem. The following example demonstrates how to decrease the MSS value via mangle:

```
[admin@Hautespot] > /ip firewall mangle add out-interface=pppoe-out \
\... protocol=tcp tcp-flags=syn action=change-mss new-mss=1300 chain=forward
[admin@Hautespot] > /ip firewall mangle print
Flags: X - disabled, I - invalid, D - dynamic
 0   chain=forward out-interface=pppoe-out protocol=tcp tcp-flags=syn
     action=change-mss new-mss=1300

[admin@Hautespot] >
```

Maximum Transmission Unit on HauteWRAP Routers

Background

It is sole responsibility of administrator to configure MTUs such that intended services and applications can be successfully implemented in network. In other words - administrator must make sure that MTUs are configured in a way that packet sizes does not exceed the capabilities of network equipment.

Originally MTU was introduced because of the high error rates and low speed of communications. Fragmentation of the data stream gives ability to correct corruption errors only by resending corrupted fragment, not the whole stream. Also on low speed

connections such as modems it can take too much time to send a big fragment, so in this case communication is possible only with smaller fragments.

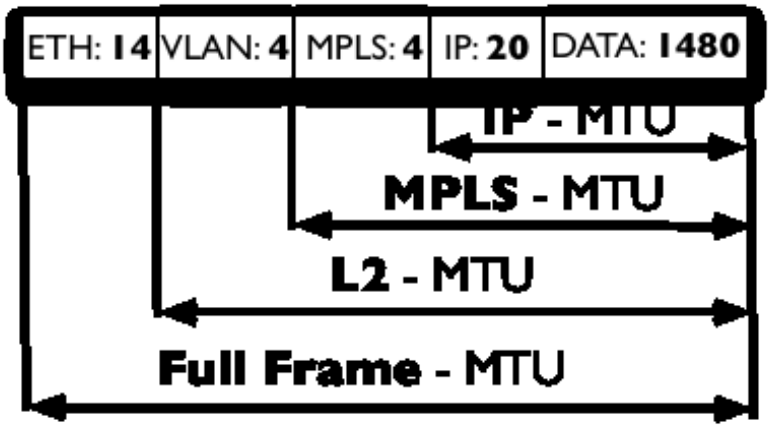
But in present days we have much lower error rates and higher speed of communication, this opens a possibility to increase the value of MTU. By increasing value of MTU we will result in less protocol overhead and reduce CPU utilization mostly due to interrupt reduction.

This way some non-standard frames started to emerge:

- **Giant** or **Jumbo** frames - frames that are bigger than standard (IEEE) Ethernet MTU
- **Baby Giant** or **Baby Jumbo** frames - frames that are just slightly bigger that standard (IEEE) Ethernet MTU

It is common now for Ethernet interfaces to support physical MTU above standard, but this can not be taken for granted. Abilities of other network equipment must be taken into account as well - for example, if 2 routers with Ethernet interfaces supporting physical MTU 1526 are connected through Ethernet switch, in order to successfully implement some application that will produce this big Ethernet frames, switch must also support forwarding such frames.

MTU on HauteRouterOS



HauteSpot HauteRouterOS recognizes several types of MTU:

- IP/Layer-3/L3 MTU
- MPLS/Layer-2.5/L2.5 MTU
- MAC/Layer-2/L2 MTU
- Full frame MTU

Full frame MTU

Full frame MTU indicates the actual size of the frame that are sent by particular interface. Frame Checksum is not included as it is removed by Ethernet driver as soon as frame reach its destination.

MAC/Layer-2/L2 MTU

L2MTU indicates maximal size of the frame without MAC header that can be sent by this interface.

Starting from the HauteRouterOS v3.25 L2MTU values can be seen in "/interface" menu. L2MTU support is added for all HauteWRAP related Ethernet interfaces, VLANs, Bridge, VPLS and wireless interfaces. Some of them support configuration of L2MTU value. All other Ethernet interfaces might indicate L2MTU only if the chip set is the same as HauteWRAP Ethernets.

This will allow users to check if desired setup is possible. Users will be able to utilize additional bytes for VLAN and MPLS tags, or simple increase of interface MTU to get rid of the some unnecessary fragmentation.

This table shows L2MTU supported by HauteSpot HauteWRAP routers:

HauteWRAP	ether1	ether2	ether3
HR-WRAPSXC **HR-WRAPDXC**	1526		
HR-WRAPSXC3E **HR-WRAPDXC3E**	1526	1522	1522
HR-WRAPLX	1526		
HR-WRAPSXB **HR-WRAPDXB**	1600		

All wireless interfaces in HauteRouterOS (including TLP Full Duplex) support 2290 byte L2MTU

MPLS/Layer-2.5/L2.5 MTU

Configured in "/mpls interface" menu, specifies maximal size of packet, including MPLS labels, that is allowed to send out by the particular interface (default is 1508).

Make sure that MPLS MTU is smaller or equal to L2MTU

MPLS MTU affects packets depending on what action MPLS router is performing. It is strongly recommended that MPLS MTU is configured to the same value on all routers forming MPLS cloud because of effects MPLS MTU has on MPLS switched packets. This requirement means that all interfaces participating in MPLS cloud must be configured to the smallest MPLS MTU values among participating interfaces, therefore care must be taken to properly select hardware to be used.

MPLS Switching

If packet with labels included is bigger than MPLS MTU, MPLS tries to guess protocol that is carried inside MPLS frame.

If this is IP packet, MPLS produces ICMP Need Fragment error. This behavior mimics IP protocol behavior. Note that this ICMP error is not routed back to originator of packet but is switched towards end of LSP, so that egress router can route it back.

If this is not IP packet, MPLS simply drops it, because it does not know how to interpret the contents of packet. This feature is very important in situations where MPLS

applications such as VPLS are used (where frames that are MPLS tagged are not IP packets, but e.g. encapsulated Ethernet frames as in case of VPLS) - if somewhere along the LSP MPLS MTU will be less than packet size prepared by ingress router, frames will simply get dropped.

IP ingress

When router first introduces label (or labels) on IP packet, and resulting packet size including MPLS labels exceeds MPLS MTU, router behaves as if interface MTU was exceeded - either fragments packet in fragments that does not exceed MPLS MTU when labels are attached (if IP Dont Fragment is not set), or generates ICMP Need Fragmentation error that is sent back to originator.

VPLS ingress

When router encapsulates Ethernet frame for forwarding over VPLS pseudowire, it checks if packet size with VPLS Control Word (4 bytes) and any necessary labels (usually 2 labels - 8 bytes), exceeds MPLS MTU of outgoing interface. If it does, VPLS fragments packet so that it honours MPLS MTU of outgoing interface. Packet is defragmented at egress point of VPLS pseudowire.

IP/Layer-3/L3 MTU

Configured as interface MTU setting (/interface <type> <name> set mtu=X). Specifies how big IP packets router is allowed to send out the particular interface.

If router receives IP packet of size 1500, but MTU for outgoing interface is set to 1400, router will either fragment the packet (if "Don't Fragment" bit is not set in IP header) or drop the packet and send ICMP "Need Fragmentation" error back to originator (this is essential for Path MTU Discovery to work).

Sometimes it can be bad idea to change IP MTU from its default 1500 bytes on router interfaces if complete path end-to-end is not in administrators control. Although IP fragmentation and end-to-end Path MTU Discovery is intended to handle this situation, if ICMP Need Fragmentation errors are filtered somewhere along the path, Path MTU Discovery will not work.

In general, it is best to leave all interfaces set to the default IP MTU of 1500 bytes. This will allow IP packets of up to 1472 bytes to pass end to end. To test a link for full MTU length handling you can use the following command under MS Windows:

```
ping –f –l 1472 192.168.1.4
```

In Linux the comparible command is:

ping –s 1472 192.168.1.4

In this example the target to test to is 192.168.1.4. If you receive back a message stating that “Packet needs to be fragmented but DF set.” then your MTU is too large for the MTU settings of the devices along the path. In Linux your packets will not pass.

Note that ping has 8 bytes of overhead so your actual packet size will be 1480 when –l is set to 1472. Packets larger than this will be fragmented with an MTU of 1500. So not running the –f command will still pass pings, but not as single packets.

While testing you should make sure that you do not have other conflicting streaming traffic, etc that may effect responses from your remote ping target. If the target is busy trying to send video, like in the case of an IP camera, it may not process ping requests timely.

There are several features in HauteSpot HauteRouterOS that can benefit from the possibility to exceed standard MTU

Simple Examples

In these examples we will take a look at frames entering and leaving router via Ethernet interfaces.

Simple Routing

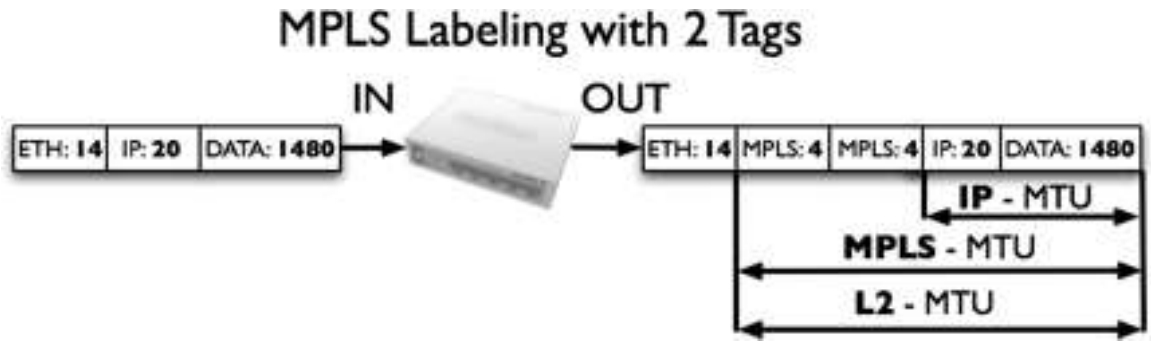
The image shows the packet MTU size for simple routing, packets size is not modified.

Routing with VLAN Encap

Each VLAN tag is 4 bytes long, VLAN tag is added by router. L2-MTU is increased by 4 bytes.

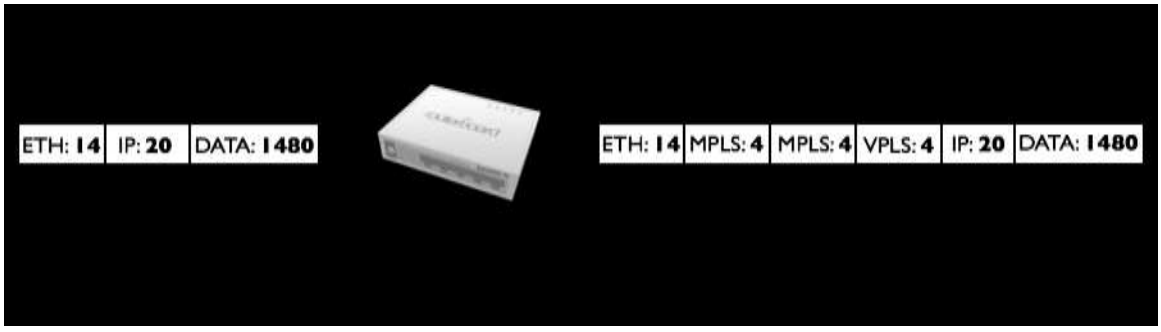
Simple MPLS with tags

When MPLS is used as plain replacement for IP routing, only one label is attached to every packet, therefore packet size increases by 4 bytes, we have the situation with two MPLS labels. In order to be able to forward standard size (1500 bytes) IP packet without fragmentation, MPLS MTU must be set to at least 1508 for two MPLS labels.



VPLS Tunnel

Two MPLS labels are present, when remote endpoint is not directly attached. One MPLS label is used to get to remote endpoint, second label is used to identify VPLS tunnel.



L2MTU advanced example

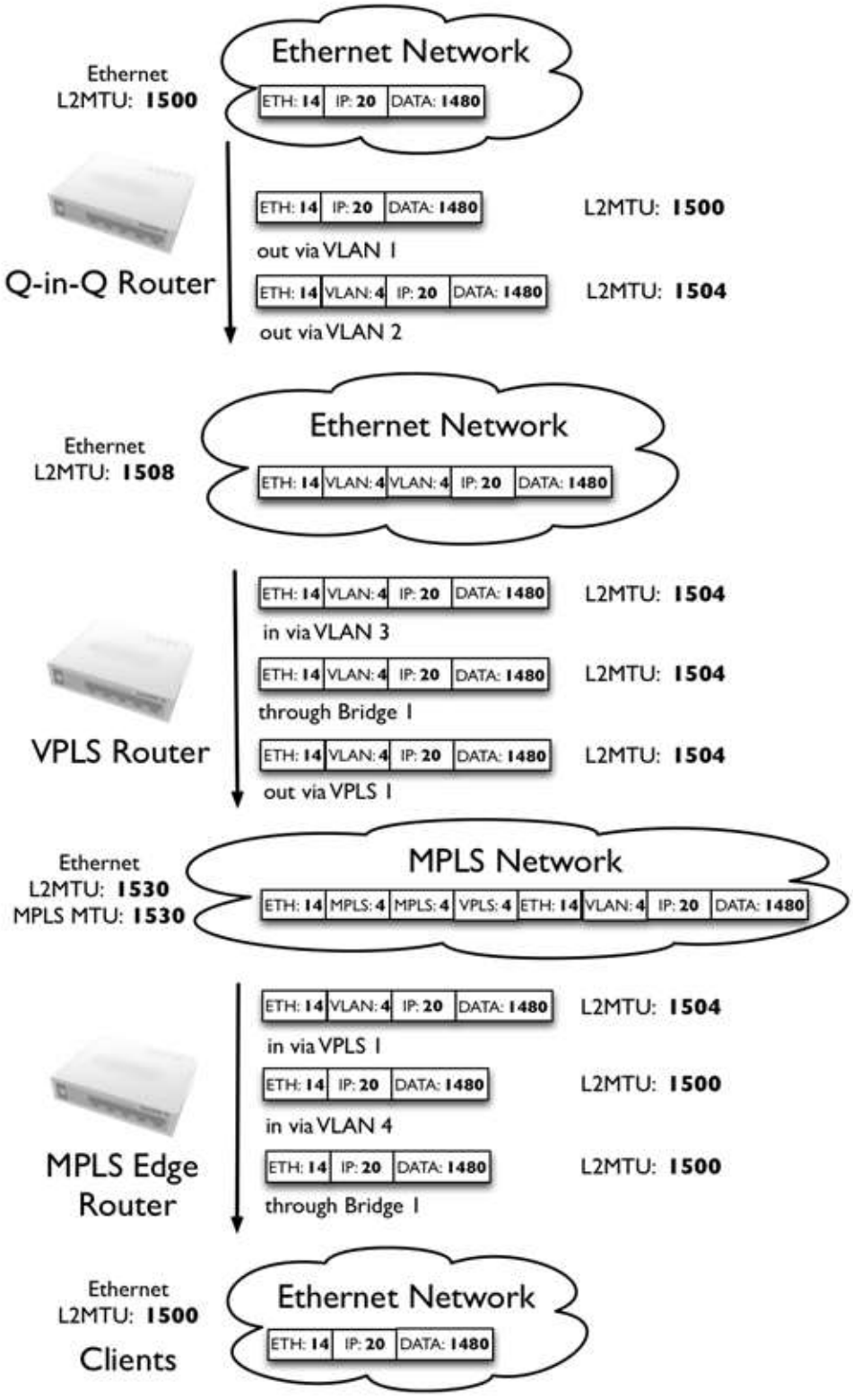
In this example we will take a closer look at required L2MTU of all Ethernet like interfaces including Bridge, VLAN, VPLS interfaces.

In this setup we will have 3 routers:

- Q-in-Q router - this router will receive standard 1500 byte Ethernet frame and will add two VLAN tags to the packet. Then packet will be sent out via Ethernet network to the second router

- VPLS router - this router will remove outer VLAN tag and will bridge packet with the remaining VLAN tag with VPLS tunnel. VPLS tunnel will take packet through the MPLS network to the third router.

- MPLS Edge router - will remove VPLS and VLAN tags and bridge packet to the client Ethernet network.


Ethernet
L2MTU: 1500
Ethernet Network
ETH: 14 IP: 20 DATA: 1480
Q-in-Q Router
ETH: 14 IP: 20 DATA: 1480
L2MTU: 1500
out via VLAN 1
ETH: 14 VLAN: 4 IP: 20 DATA: 1480
L2MTU: 1504
out via VLAN 2
Ethernet
L2MTU: 1508
Ethernet Network
ETH: 14 VLAN: 4 VLAN: 4 IP: 20 DATA: 1480
VPLS Router
ETH: 14 VLAN: 4 IP: 20 DATA: 1480
L2MTU: 1504
in via VLAN 3
ETH: 14 VLAN: 4 IP: 20 DATA: 1480
L2MTU: 1504
through Bridge 1
ETH: 14 VLAN: 4 IP: 20 DATA: 1480
L2MTU: 1504
out via VPLS 1
Ethernet
L2MTU: 1530
MPLS MTU: 1530
MPLS Network
ETH: 14 MPLS: 4 MPLS: 4 VPLS: 4 ETH: 14 VLAN: 4 IP: 20 DATA: 1480
MPLS Edge Router
ETH: 14 VLAN: 4 IP: 20 DATA: 1480
L2MTU: 1504
in via VPLS 1
ETH: 14 IP: 20 DATA: 1480
L2MTU: 1500
in via VLAN 4
ETH: 14 IP: 20 DATA: 1480
L2MTU: 1500
through Bridge 1
Ethernet
L2MTU: 1500
Clients
Ethernet Network
ETH: 14 IP: 20 DATA: 1480

MIMO Setup Guide

Brief Overview

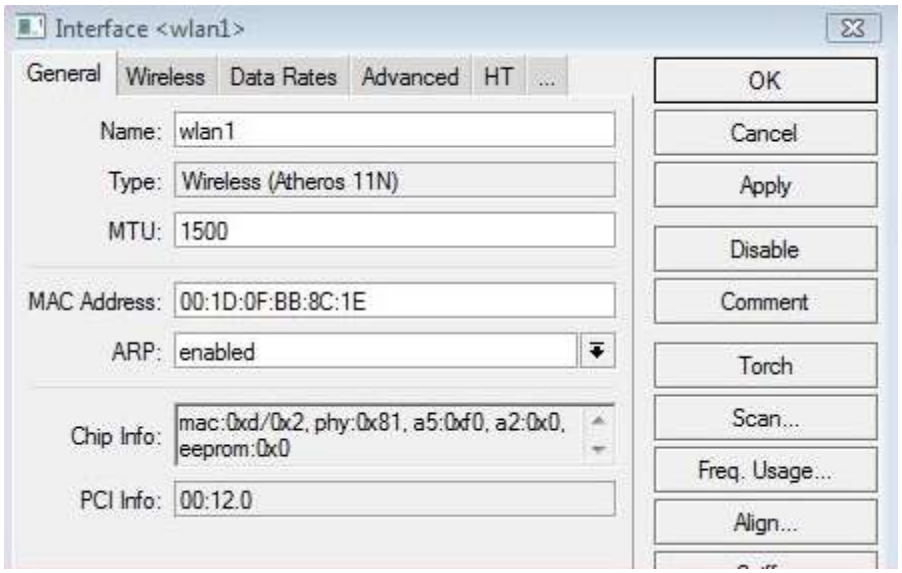
Since the release of HauteRouterOS v4.03beta, 802.11n (MIMO) draft wireless cards are now supported. This guide's purpose is to explain to those familiar with the use of regular wireless, exactly how to go about setting up an MIMO interface.

Pre-Setup

Equipment

Equipment used on my test bench:

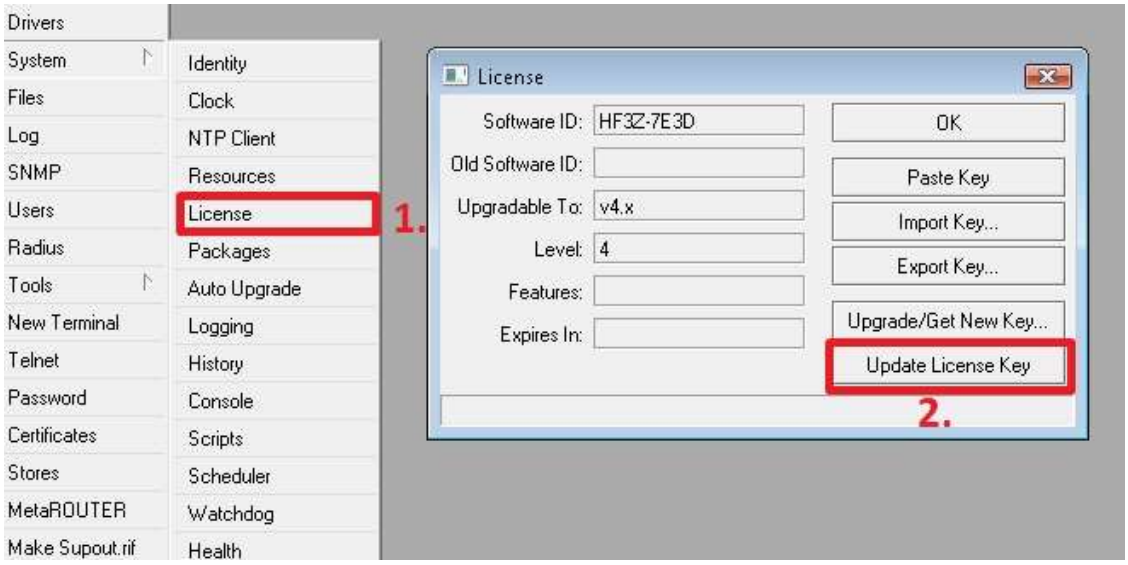
- HR-WRAPSXC-N x 2



Updating License key and unlocking 802.11n

Once you have completed the upgrade to v4, connect via HSConfig to your device then:

1. Open the license page via System -> License
2. Click on the "update license key"



Setup and Testing

This section details how to configure your wireless MIMO interface along with an example at the end of a preconfigured setup you can attempt on your own cards. I will also go into as much detail as possible on the new options provided to the MIMO interfaces and what each of these does.

MIMO Configuration options

HSConfig changeable configuration options

- **band** (2ghz-b/g/n | 2ghz-onlyn | 5ghz-a/n | 5ghz-onlyn | other-original a/b/g options also available)
- **ht-extension-channel** (above-control | below-control | disabled)
- **ht-rxchains** (0,1,2 - any combination of these)
- **ht-txchains** (0,1,2 - any combination of these)
- **ht-ampdu-priorities** (0,1,2,3,4,5,6,7 - any combination of these)
- **ht-guard-interval** (any | long)

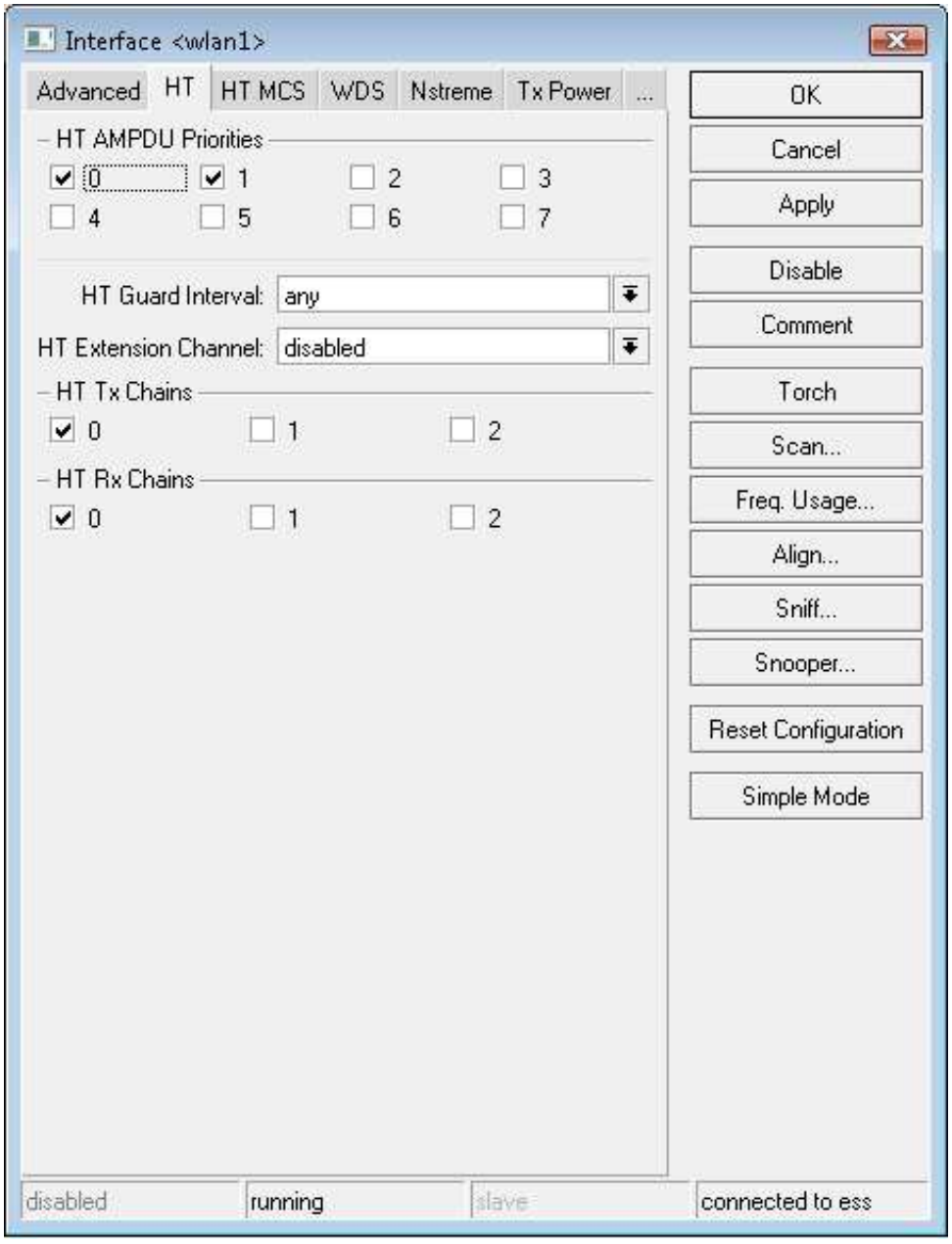
These items can be viewed and changed from in HSConfig or terminal.

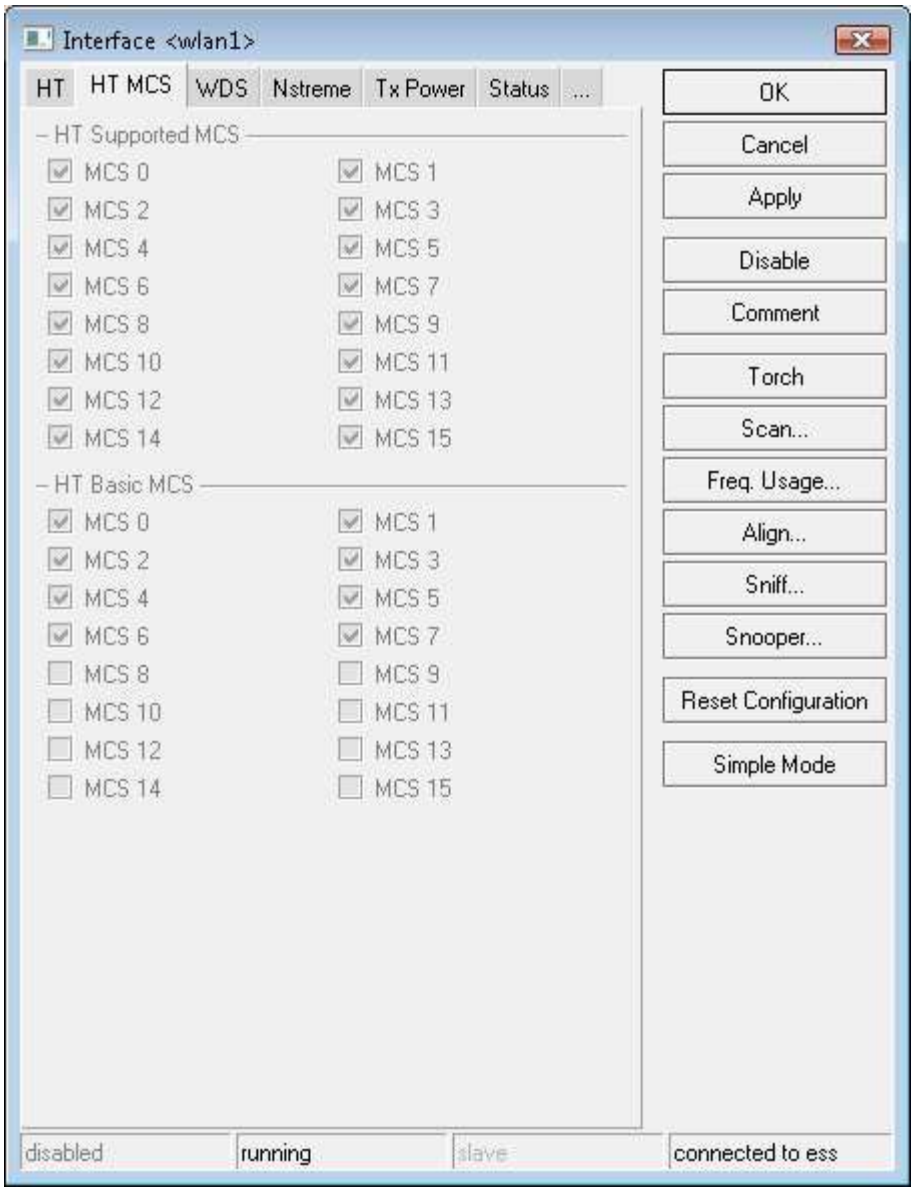
HSConfig viewable configuration options

- **ht-basic-mcs** (mcs-0,mcs-1,mcs-2,mcs-3,mcs-4,mcs-5,mcs-6,mcs-7,mcs-8,mcs-9,mcs-10,mcs-11,mcs-12,mcs-13,mcs-14,mcs-15 - any combination of these)
- **ht-supported-mcs** (mcs-0,mcs-1,mcs-2,mcs-3,mcs-4,mcs-5,mcs-6,mcs-7,mcs-8,mcs-9,mcs-10,mcs-11,mcs-12,mcs-13,mcs-14,mcs-15 - any combination of these)

These previous 2 options can be changed only from terminal, but appear in HSConfig. (In ROS4.0beta4 they can be changed in HSConfig Wireless Interface window, tab "Data Rates" and set rate to "configured". Under "HT MCS" tab options are now configurable.)

HSConfig MIMO configuration tabs





Configurable options, in depth.

- **ht-extension-channel** (above-control | below-control | disabled)

The current 802.11n draft supports a method of channel bonding for both 2.4Ghz and 5.Ghz systems. There is 1 20Mhz channel defined as the "control channel" while the secondary channel can then be set to sit above or below the control channel. Seeing as 2.4Ghz only has 3 'usable' channels (1,6,11) its recommended to use this on a 5Ghz where there are more available channels.

- **ht-rxchains** (0,1,2 - any combination of these)

- **ht-txchains** (0,1,2 - any combination of these)

Which antenna connector to use for TX or RX. You can use one of these or multiple (depending on your antenna configuration). On HauteSpot MIMO cards there are 2 antenna connectors and to use both of them ht-tx/rx-chains should be set to 0,1.

- **ht-ampdu-priorities** (0,1,2,3,4,5,6,7 - any combination of these)

A-MPDU (Aggregated Mac Protocol Data Unit) allows the transmissions of multiple ethernet frames to a single location as burst of up to 64kbytes This is performed on the hardware itself. It is important to understand these numbers, 0-7 correspond to packet priority for which A-MPDU will be used. For example, if you check options 0 and 6, A-MPDU will only be used on packets with a priority of 0 and 6, no other packets will be affected. A-MPDU should not be used with applications that require low latency such as VOIP since the price paid for higher throughput is increased latency.

- **ht-guard-interval** (any | long)

In 802.11n the OFDM Guard interval (GI) is decreased from 800ns to 400ns, without any further info I am assuming that the 'any' is the new 400ns timing and 'long' is the old 800ns

Example configuration

As above I used 2 x RB411's HR-WRAPSXC-N for my test setup, here is a sample config's for each one.

Access Point

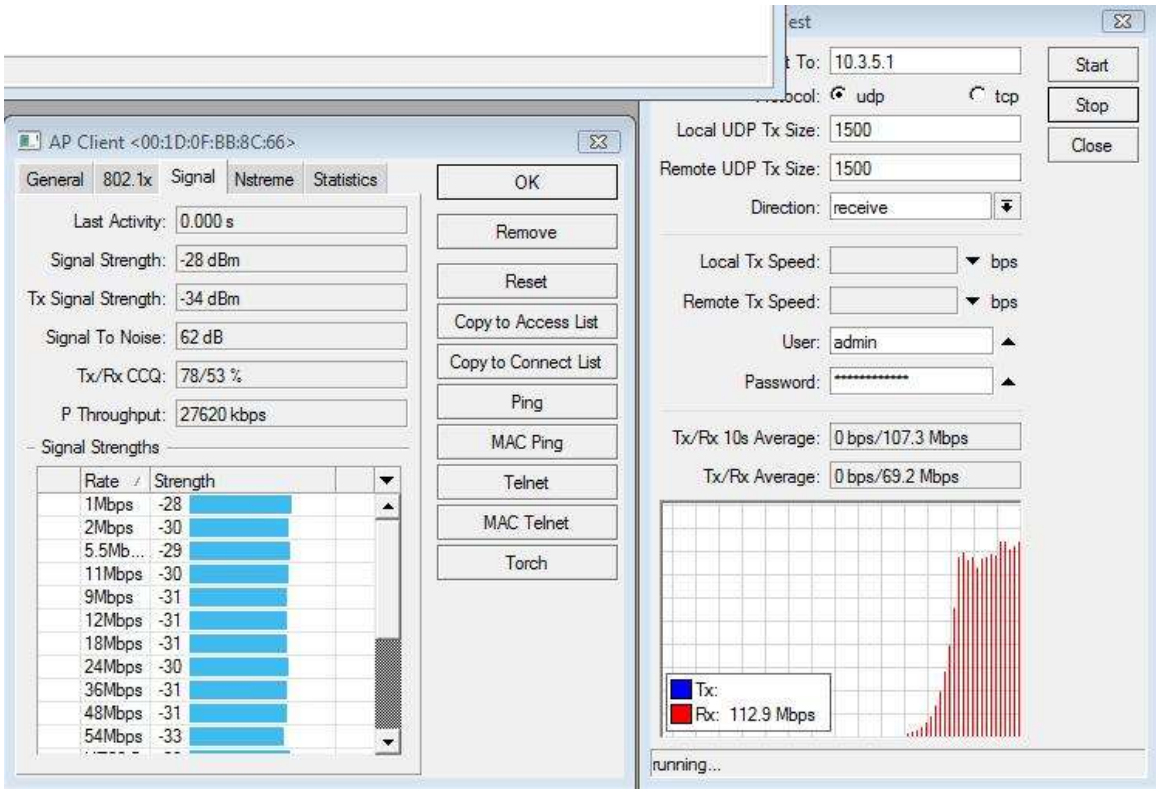
```
/ip address
add address=192.168.1.1/24 broadcast=192.168.1.255 comment=""
disabled=no interface=wlan1 network=192.168.1.0
/interface wireless
set 0 ack-timeout=dynamic adaptive-noise-immunity=none allow-
sharedkey=no antenna-gain=0 antenna-mode=ant-a area="" arp=enabled
band=2ghz-b/g/n basic-rates-a/g=6Mbps \
    basic-rates-b=1Mbps comment="" compression=no
country=no_country_set default-ap-tx-limit=0 default-authentication=yes
default-client-tx-limit=0 default-forwarding=\
    yes dfs-mode=none disable-running-check=no disabled=no disconnect-
timeout=3s frame-lifetime=0 frequency=2412 frequency-mode=manual-
txpower hide-ssid=no \
    ht-ampdu-priorities=0 ht-amsdu-limit=8192 ht-amsdu-threshold=8192
ht-basic-mcs=mcs-0,mcs-1,mcs-2,mcs-3,mcs-4,mcs-5,mcs-6,mcs-7 ht-
extension-channel=above-control \
```

```
    ht-guard-interval=any ht-rxchains=0,2 ht-supported-mcs=mcs-0,mcs-
1,mcs-2,mcs-3,mcs-4,mcs-5,mcs-6,mcs-7,mcs-8,mcs-9,mcs-10,mcs-11,mcs-
12,mcs-13,mcs-14,mcs-15 \
    ht-txchains=0,2 hw-fragmentation-threshold=disabled hw-protection-
mode=none hw-protection-threshold=0 hw-retries=4 mac-
address=00:1D:0F:BB:8C:66 max-station-count=\
    2007 mode=ap-bridge mtu=1500 name=wlan1 periodic-
calibration=default periodic-calibration-interval=60 preamble-mode=both
\
    proprietary-extensions=post-2.9.25 radio-name=001D0FBB8C66 rate-
set=default scan-list=default security-profile=default ssid=abc
station-bridge-clone-mac=\
    00:00:00:00:00:00 supported-rates-
a/g=6Mbps,9Mbps,12Mbps,18Mbps,24Mbps,36Mbps,48Mbps,54Mbps supported-
rates-b=1Mbps,2Mbps,5.5Mbps,11Mbps tx-power-mode=default \
    wds-mode=disabled wmm-support=enabled
```

Wireless Station

```
/ip address
add address=192.168.1.2/24 broadcast=192.168.1.255 comment=""
disabled=no interface=wlan1 network=192.168.1.0
/interface wireless
set 0 ack-timeout=dynamic adaptive-noise-immunity=none allow-
sharedkey=no antenna-gain=0 antenna-mode=ant-a area="" arp=enabled
band=2ghz-b/g/n basic-rates-a/g=6Mbps \
    basic-rates-b=1Mbps comment="" compression=no
country=no_country_set default-ap-tx-limit=0 default-authentication=yes
default-client-tx-limit=0 default-forwarding=\
    yes dfs-mode=none disable-running-check=no disabled=no disconnect-
timeout=3s frame-lifetime=0 frequency=2412 frequency-mode=manual-
txpower hide-ssid=no \
    ht-ampdu-priorities=0 ht-amsdu-limit=8192 ht-amsdu-threshold=8192
ht-basic-mcs=mcs-0,mcs-1,mcs-2,mcs-3,mcs-4,mcs-5,mcs-6,mcs-7 ht-
extension-channel=above-control \
    ht-guard-interval=any ht-rxchains=0,2 ht-supported-mcs=mcs-0,mcs-
1,mcs-2,mcs-3,mcs-4,mcs-5,mcs-6,mcs-7,mcs-8,mcs-9,mcs-10,mcs-11,mcs-
12,mcs-13,mcs-14,mcs-15 \
    ht-txchains=0,2 hw-fragmentation-threshold=disabled hw-protection-
mode=none hw-protection-threshold=0 hw-retries=4 mac-
address=00:1D:0F:BB:8C:1E max-station-count=\
    2007 mode=station mtu=1500 name=wlan1 periodic-calibration=default
periodic-calibration-interval=60 preamble-mode=both \
    proprietary-extensions=post-2.9.25 radio-name=001D0FBB8C1E rate-
set=default scan-list=default security-profile=default ssid=abc
station-bridge-clone-mac=\
    00:00:00:00:00:00 supported-rates-
a/g=6Mbps,9Mbps,12Mbps,18Mbps,24Mbps,36Mbps,48Mbps,54Mbps supported-
rates-b=1Mbps,2Mbps,5.5Mbps,11Mbps tx-power-mode=default \
    wds-mode=disabled wmm-support=enabled
```

My first test gave similar to the following results, thou I expect to improve on these shortly.



Current Optimum setup configuration

```
/interface wireless security-profiles
set default authentication-types="" eap-methods=passthrough group-
ciphers="" \
    group-key-update=5m interim-update=0s management-
protection=disabled \
    management-protection-key="" mode=none name=default \
    radius-eap-accounting=no radius-mac-accounting=no \
    radius-mac-authentication=no radius-mac-caching=disabled \
    radius-mac-format=XX:XX:XX:XX:XX:XX radius-mac-mode=as-username \
    static-algo-0=none static-algo-1=none static-algo-2=none static-
algo-3=\
    none static-key-0="" static-key-1="" static-key-2="" static-key-
3="" \
    static-sta-private-algo=none static-sta-private-key="" \
    static-transmit-key=key-0 supplicant-identity=HauteSpot tls-
certificate=\
    none tls-mode=no-certificates unicast-ciphers="" wpa-pre-shared-
key="" \
    wpa2-pre-shared-key=""
/interface wireless
set 0 ack-timeout=dynamic adaptive-noise-immunity=none allow-
sharedkey=no \
    antenna-gain=0 antenna-mode=ant-a area="" arp=enabled band=5ghz-
onlyn \
```

```
    basic-rates-a/g="" basic-rates-b="" comment="" compression=no
country=\
    no_country_set default-ap-tx-limit=0 default-authentication=yes \
    default-client-tx-limit=0 default-forwarding=yes dfs-mode=none \
    disable-running-check=no disabled=no disconnect-timeout=3s \
    frame-lifetime=0 frequency=5180 frequency-mode=manual-txpower hide-
ssid=\
    no ht-ampdu-priorities=0,1,2,3,4,5,6,7 ht-amsdu-limit=8192 \
    ht-amsdu-threshold=8192 ht-basic-mcs=mcs-0 ht-extension-
channel=disabled \
    ht-guard-interval=any ht-rxchains=0,1 ht-supported-mcs=\
    mcs-0,mcs-7,mcs-8,mcs-14,mcs-15 ht-txchains=0,1 \
    hw-fragmentation-threshold=disabled hw-protection-mode=none \
    hw-protection-threshold=0 hw-retries=15 mac-
address=00:0C:42:3A:CF:E9 \
    max-station-count=2 mode=bridge mtu=1500 name=wlan1 on-fail-retry-
time=\
    100ms periodic-calibration=disabled periodic-calibration-
interval=60 \
    preamble-mode=short proprietary-extensions=post-2.9.25 radio-name=\
    000C423ACFE9 rate-set=configured scan-list=5180,5760,5320,5825 \
    security-profile=default ssid=awmn-2315-3665 station-bridge-clone-
mac=\
    00:00:00:00:00:00 supported-rates-a/g="" supported-rates-b="" tx-
power=3 \
    tx-power-mode=all-rates-fixed update-stats-interval=disabled \
    wds-cost-range=50-150 wds-default-bridge=none wds-default-cost=100
\
    wds-ignore-ssid=no wds-mode=disabled wmm-support=enabled
/interface wireless manual-tx-power-table
set wlan1 comment="" manual-tx-
powers="1Mbps:17,2Mbps:17,5.5Mbps:17,11Mbps:17,\

6Mbps:17,9Mbps:17,12Mbps:17,18Mbps:17,24Mbps:17,36Mbps:17,48Mbps:17,54M
bps\
    :17,HT20-1:0,HT20-2:0,HT20-3:0,HT20-4:0,HT20-5:0,HT20-6:0,HT20-
7:0,HT20-8:\
    0,HT40-1:0,HT40-2:0,HT40-3:0,HT40-4:0,HT40-5:0,HT40-6:0,HT40-
7:0,HT40-8:0"
/interface wireless nstreme
set wlan1 comment="" disable-csma=no enable-nstreme=yes enable-
polling=yes \
    framer-limit=3998 framer-policy=best-fit
/interface wireless align
set active-mode=yes audio-max=-20 audio-min=-100 audio-monitor=\
    00:00:00:00:00:00 filter-mac=00:00:00:00:00:00 frame-size=300 \
    frames-per-second=25 receive-all=no ssid-all=no
/interface wireless sniffer
set channel-time=200ms file-limit=10 file-name="" memory-limit=10 \
    multiple-channels=no only-headers=no receive-errors=no streaming-
enabled=\
    no streaming-max-rate=0 streaming-server=0.0.0.0
/interface wireless snooper
set channel-time=200ms multiple-channels=yes receive-errors=no
```

MPLS Overview

MPLS stands for MultiProtocol Label Switching. It kind of replaces IP routing - packet forwarding decision (outgoing interface and next hop router) is no longer based on fields in IP header (usually destination address) and routing table, but on labels that are attached to packet. This approach speeds up forwarding process because next hop lookup becomes very simple compared to routing lookup (finding longest matching prefix).

Efficency of forwarding process is the main benefit of MPLS, but it must be taken into account that MPLS forwarding disables processing of network layer (e.g. IP) headers, therefore no network layer based actions like NAT and filtering can be applied to MPLS forwarded packets. Any network layer based actions should be taken on ingress or egress of MPLS cloud, with preferred way being ingress - this way, e.g. traffic that is going to be dropped anyway does not travel through MPLS backbone.

In the simplest form MPLS can be thought of like improved routing - labels are distributed by means of LDP protocol for routes that are active and labelled packet takes the same path it would take if it was not labelled. Router that routes unlabelled packet using some route for which it has received label from next hop, imposes label on packet and send it to next hop - it gets MPLS switched further along its path. Router that receives packet with label it has assigned to some route changes packet label with one received from next hop of particular route and sends packet to next hop. Label switched path ensures delivery of data to the MPLS cloud egress point. Applications of MPLS are based on this basic MPLS concept of label switched paths.

Another way of establishing label switching path is traffic engineering tunnels (TE tunnels) by means of RSVP TE protocol. Traffic engineering tunnels allow explicitly routed LSPs and constraint based path selection (where constraints are interface properties and available bandwidth).

Taking into account complexity, new protocols and applications that MPLS introduces and differences of concepts that MPLS adds to routed/bridged network, it is recommended to have in depth understanding of MPLS concepts before implementing MPLS in production network. Some suggested reading material:

- Multiprotocol Label Switching http://en.wikipedia.org/wiki/Multiprotocol_Label_Switching
- RFC3031 Multiprotocol Label Switching Architecture http://www.ietf.org/rfc/rfc3031.txt
- MPLS Fundamentals by Luc De Ghein http://www.amazon.com/MPLS-Fundamentals-Luc-Ghein/dp/1587051974

HauteRouterOS MPLS features

As of version 3.8 MPLS feature development for HauteRouterOS continues in **mpls-test** package that requires **routing-test** package. Currently HauteRouterOS (by means of mpls-test and routing-test packages) supports the following MPLS related features:

- MPLS switching with penultimate hop popping support
- static local label bindings for IPv4
- static remote label bindings for IPv4
- Label Distribution Protocol (RFC 3036, RFC 5036) for IPv4
    - downstream unsolicited label advertisement
    - independent label distribution control
    - liberal label retention
    - targeted session establishment
    - optional loop detection
- Virtual Private Lan Service
    - VPLS LDP signalling (RFC 4762)
    - VPLS pseudowire fragmentation and reassembly (RFC 4623)
    - VPLS MP-BGP based autodiscovery and signaling (RFC 4761), see BGP based VPLS
- RSVP TE Tunnels
    - tunnel head-end
    - explicit paths
    - OSPF extensions for TE tunnels
    - CSPF path selection
    - forwarding of VPLS and MPLS IP VPN traffic on TE tunnels
- MP-BGP based MPLS IP VPN
- OSPF extensions for MPLS TE

Features since version 3.17:

- support for OSPF as CE-PE protocol
- ping and traceroute for specified VRF
- control over network layer TTL propagation in MPLS

Features since version 3.20 (note that this version changes configuration syntax and adds new parameters!):

- Cisco style static VPLS pseudowires (RFC 4447 FEC type 0x80)
- Cisco VPLS BGP-based auto-discovery (draft-ietf-l2vpn-signaling-08)
- support for multiple import/export route target extended communities for BGP based VPLS (both, RFC 4761 and draft-ietf-l2vpn-signaling-08)

Features since version 3.23

- Ingress TE tunnel rate limit and automatic reserved bandwidth adjustment.

- all tunnel bandwidth settings are specified and displayed in bits per second
- complete support for OSPF as PE-CE routing protocol (including sham links)

Features since version 3.24

- RIP as CE-PE protocol
- per-VRF BGP instance redistribution settings

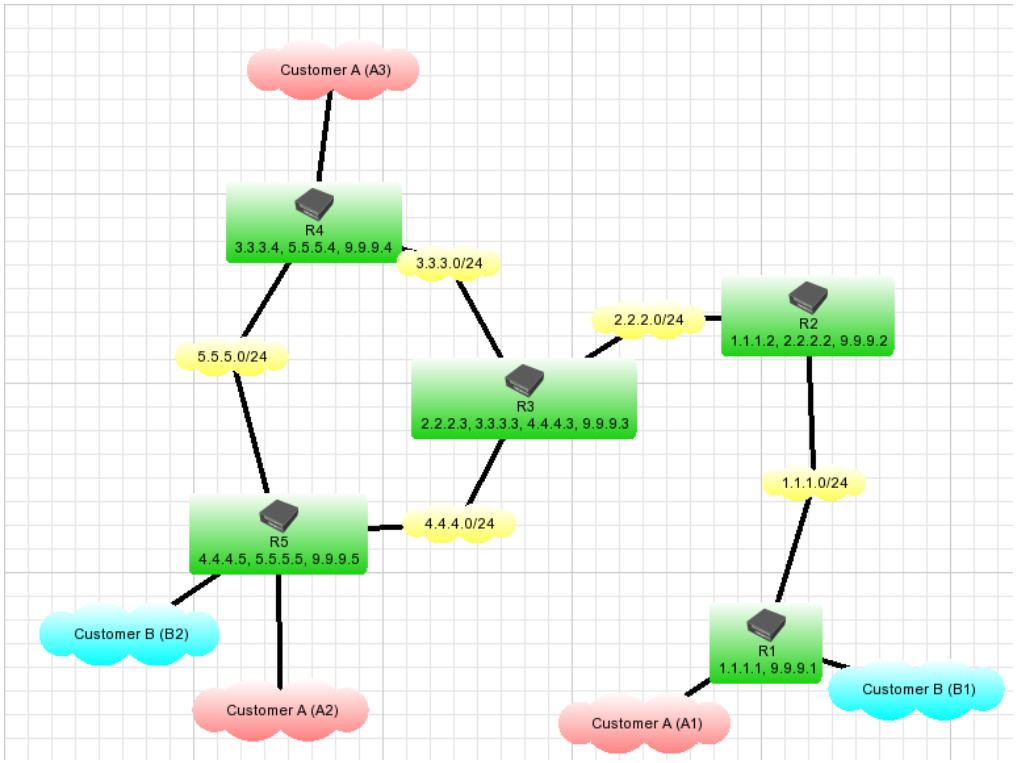
MPLS features that HauteRouterOS DOES NOT HAVE yet:

- IPv6 support
- LDP features:
  - downstream on demand label advertisement
  - ordered label distribution control
  - conservative label retention
- MPLS IP VPN features
  - IBGP as CE-PE protocol support
- TE features
  - fast reroute
  - link/node protection

To ensure compatibility with other manufacturer equipment ensure that required features match, if uncertain, consult with HauteSpot support. RouterOS LDP and TE implementation has been tested with Cisco IOS.

Example network

Consider network service provider that is connecting 3 remote sites of Customer A (A1,A2 and A3) and 2 remote sites of Customer B (B1 and B2) using its routed IP network core, consisting of routers (R1-R5):



Customers require transparent ethernet segment connection between sites. So far it has been implemented by means of bridging EoIP tunnels with physical ethernet interfaces.
Note that there are no IP addresses configured on R1, R4 and R5 interfaces that face customer networks.
Enabling MPLS forwarding can speed up packet forwarding process in such network. Using one of MPLS applications - VPLS can further increase efficency of ethernet frame forwarding by not having to encapsulate ethernet frames in IP frames, thus removing IP header overhead.
This guide gives step by step instructions that will lead to implementation of VPLS to achieve necessary service.

Prerequisites for MPLS

"Loopback" IP address

Although not a strict requirement, it is advisable to configure routers participating in MPLS network with "loopback" IP addresses (not attached to any real network interface) to be used by LDP to establish sessions.
This serves 2 purposes:

- as there is only one LDP session between any 2 routers, no matter how many links connect them, loopback IP address ensures that LDP session is not affected by interface state or address changes
- use of loopback address as LDP transport address ensures proper penultimate hop popping behaviour when multiple labels are attached to packet as in case of VPLS

In RouterOS "loopback" IP address can be configured by creating dummy bridge interface without any ports and adding address to it. For example, on R1 it is done with the following commands:

```
/interface bridge add name=lobridge
/ip address add address=9.9.9.1/32 interface=lobridge
```

The rest of routers are configured similar way.

IP connectivity

As LDP distributes labels for active routes, essential requirement is properly configured IP routing. LDP by default distributes labels for active IGP routes (that is - connected, static, and routing protocol learned routes, except BGP).
In given example setup OSPF is used to distribute routes. For example, on R5 OSPF is configured with the following commands:

```
/routing ospf set redistribute-connected=as-type-1
/routing ospf network add area=backbone network=4.4.4.0/24
/routing ospf network add area=backbone network=5.5.5.0/24
```

On other routers OSPF is configured in similar way.
This yields routing table on R5 like this:

```
[admin@R5] > ip route print
Flags: X - disabled, A - active, D - dynamic, C - connect, S -
static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        GATEWAY-STATE GATEWAY
DISTANCE             INTERFACE
 0 ADo  1.1.1.0/24                         reachable     4.4.4.3
110                  ether1
 1 ADo  2.2.2.0/24                         reachable     4.4.4.3
110                  ether1
 2 ADo  3.3.3.0/24                         reachable     4.4.4.3
110                  ether1
                                           reachable     5.5.5.4
ether2
 3 ADC  4.4.4.0/24        4.4.4.5
0                    ether1
 4 ADC  5.5.5.0/24        5.5.5.5
0                    ether2
 5 ADo  9.9.9.1/32                         reachable     4.4.4.3
110                  ether1
 6 ADo  9.9.9.2/32                         reachable     4.4.4.3
110                  ether1
 7 ADo  9.9.9.3/32                         reachable     4.4.4.3
110                  ether1
 8 ADo  9.9.9.4/32                         reachable     5.5.5.4
110                  ether2
 9 ADC  9.9.9.5/32        9.9.9.5
0                    lobridge
```

and traceroute from R5 to R1 like this:

```
[admin@R5] > /tool traceroute 9.9.9.1
```

```
     ADDRESS                                          STATUS
   1          4.4.4.3 11ms 1ms 4ms
   2          2.2.2.2 23ms 3ms 2ms
   3          9.9.9.1 26ms 3ms 3ms
```

Configuring LDP

In order to distribute labels for routes, LDP should get enabled. On R1 this is done by commands (interface ether3 is facing network 1.1.1.0/24):

```
/mpls ldp set enabled=yes transport-address=9.9.9.1 lsr-
id=9.9.9.1
/mpls ldp interface add interface=ether3
```

Note that transport-address gets set to 9.9.9.1. This makes router originate LDP session connections with this address and also advertise this address as transport address to LDP neighbors.
Other routers are configured in similar way - LDP is enabled on interfaces that connect routers and not enabled on interfaces that connect customer networks. For example, on R5:

```
[admin@R5] > /ip address print
Flags: X - disabled, I - invalid, D - dynamic
 #   ADDRESS            NETWORK         BROADCAST
INTERFACE
 0   4.4.4.5/24         4.4.4.0         4.4.4.255       ether1
 1   5.5.5.5/24         5.5.5.0         5.5.5.255       ether2
 2   9.9.9.5/32         9.9.9.5         9.9.9.5         lobridge
[admin@R5] > /mpls ldp interface print
Flags: I - invalid, X - disabled
 #   INTERFACE
HELLO-INTERVAL       HOLD-TIME
 0   ether1                                                 5s
15s
 1   ether2                                                 5s
15s
```

After LDP sessions are established, on R5 there are 2 LDP neighbors:

```
[admin@R5] > /mpls ldp neighbor print
Flags: X - disabled, D - dynamic, O - operational, T - sending-
targeted-hello, V - vpls
 #      TRANSPORT       LOCAL-TRANSPORT PEER
SEND-TARGETED ADDRESSES
 0 DO   9.9.9.4         9.9.9.5         9.9.9.4:0
no            3.3.3.4

5.5.5.4

9.9.9.4
 1 DO   9.9.9.3         9.9.9.5         9.9.9.3:0
no            2.2.2.3
```

```
3.3.3.3

4.4.4.3

9.9.9.3
```

/mpls local-bindings shows labels that this router has assigned to routes and peers it has distributed the label to. It shows that R5 has distributed labels for all its routes to both of its neighbors - R3 and R4

```
[admin@R5] > /mpls local-bindings print
Flags: X - disabled, A - advertised, D - dynamic, L - local-
route, G - gateway-route, e - egress
 #      DST-ADDRESS        LABEL      PEERS
 0 ADLe 4.4.4.0/24         impl-null  9.9.9.4:0
                                      9.9.9.3:0
 1 ADLe 9.9.9.5/32         impl-null  9.9.9.4:0
                                      9.9.9.3:0
 2 ADG  9.9.9.4/32         17         9.9.9.4:0
                                      9.9.9.3:0
 3 ADLe 5.5.5.0/24         impl-null  9.9.9.4:0
                                      9.9.9.3:0
 4 ADG  1.1.1.0/24         18         9.9.9.4:0
                                      9.9.9.3:0
 5 ADG  2.2.2.0/24         19         9.9.9.4:0
                                      9.9.9.3:0
 6 ADG  9.9.9.1/32         20         9.9.9.4:0
                                      9.9.9.3:0
 7 ADG  9.9.9.2/32         21         9.9.9.4:0
                                      9.9.9.3:0
 8 ADG  9.9.9.3/32         22         9.9.9.4:0
                                      9.9.9.3:0
 9 ADG  3.3.3.0/24         23         9.9.9.4:0
                                      9.9.9.3:0
```

/mpls remote-bindings shows labels that are allocated for routes by neighboring routers and advertised to this router:

```
[admin@R5] > /mpls remote-bindings print
Flags: X - disabled, A - active, D - dynamic
 #    DST-ADDRESS        NEXTHOP         LABEL      PEER
 0  D 4.4.4.0/24                         16         9.9.9.4:0
 1 AD 3.3.3.0/24         5.5.5.4         impl-null  9.9.9.4:0
 2  D 9.9.9.5/32                         17         9.9.9.4:0
 3 AD 9.9.9.4/32         5.5.5.4         impl-null  9.9.9.4:0
 4  D 5.5.5.0/24                         impl-null  9.9.9.4:0
 5  D 1.1.1.0/24                         18         9.9.9.4:0
 6  D 2.2.2.0/24                         19         9.9.9.4:0
 7  D 9.9.9.1/32                         20         9.9.9.4:0
```

```
 8  D 9.9.9.2/32                          21          9.9.9.4:0
 9  D 9.9.9.3/32                          22          9.9.9.4:0
10 AD 1.1.1.0/24          4.4.4.3         16          9.9.9.3:0
11 AD 2.2.2.0/24          4.4.4.3         impl-null   9.9.9.3:0
12  D 4.4.4.0/24                          impl-null   9.9.9.3:0
13  D 3.3.3.0/24                          impl-null   9.9.9.3:0
14 AD 9.9.9.1/32          4.4.4.3         17          9.9.9.3:0
15 AD 9.9.9.3/32          4.4.4.3         impl-null   9.9.9.3:0
16  D 9.9.9.4/32                          18          9.9.9.3:0
17  D 5.5.5.0/24                          19          9.9.9.3:0
18 AD 9.9.9.2/32          4.4.4.3         20          9.9.9.3:0
19  D 9.9.9.5/32                          21          9.9.9.3:0
```

Here we can observe that R5 has received label bindings for all routes from both its neighbors - R3 and R4, but only the ones for whom particular neighbor is next hop are active. For example:

```
[admin@R5] > /ip route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        G GATEWAY
DISTANCE             INTERFACE
...
 5 ADo  9.9.9.1/32                         r 4.4.4.3         110
ether1
...
[admin@R5] > /mpls remote-bindings print
Flags: X - disabled, A - active, D - dynamic
 #    DST-ADDRESS        NEXTHOP         LABEL      PEER
...
 7  D 9.9.9.1/32                          20          9.9.9.4:0
...
14 AD 9.9.9.1/32          4.4.4.3         17          9.9.9.3:0
...
```

From the above we see that R3, which is next hop for network 9.9.9.1/32 from R5 perspective, has assigned label 17 for traffic going to 9.9.9.1/32. This implies that when R5 will be routing traffic to this network, will impose label 17.
Label switching rules can be seen in /mpls forwarding-table. For example, on R3 it looks like this:

```
[admin@R3] > /mpls forwarding-table print
 # IN-LABEL            OUT-LABELS           DESTINATION
INTERFACE            NEXTHOP
 ...
 2 17                   17                   9.9.9.1/32
ether1               2.2.2.2
 ...
```

This rule says that R3 receiving packet with label 17 will change it to label 17 assigned by R2 for network 9.9.9.1/32 (R2 is next hop for 9.9.9.1/32 from R3 perspective):

```
[admin@R2] > /mpls local-bindings print
Flags: X - disabled, A - advertised, D - dynamic, L - local-
route, G - gateway-route, e - egress
 #      DST-ADDRESS        LABEL      PEERS
 ...
 3 ADG  9.9.9.1/32         17         9.9.9.1:0
                                      9.9.9.3:0
 ...
```

R2 MPLS forwarding table tells:

```
[admin@R2] > /mpls forwarding-table print
 # IN-LABEL             OUT-LABELS           DESTINATION
INTERFACE            NEXTHOP
 ...
 1 17                                        9.9.9.1/32
ether1               1.1.1.1
 ...
```

Notice, that forwarding rule does not have any out-labels. The reason for this is that R2 is doing penultimate hop popping for this network. R1 does not assign any real label for 9.9.9.1/32 network, because it is known that R1 is egress point for 9.9.9.1/32 network (router is egress point for networks that are directly connected to it, because next hop for traffic is not MPLS router), therefore is advertises "implicit null" label for this route:

```
[admin@R2] > /mpls remote-bindings print
Flags: X - disabled, A - active, D - dynamic
 #    DST-ADDRESS        NEXTHOP         LABEL      PEER
 ...
13 AD 9.9.9.1/32         1.1.1.1         impl-null  9.9.9.1:0
 ...
```

This tells R2 to forward traffic for 9.9.9.1/32 to R1 unlabelled, which is exactly what R2 mpls forwarding-table entry tells. Penultimate hop popping ensures that routers do not have to do unnecessary label lookup when it is known in advance that router will have to route packet.

Using traceroute in MPLS networks

RFC4950 introduces extensions to ICMP protocol for MPLS. The basic idea is that some ICMP messages may carry MPLS "label stack object" (list of labels that were on packet when it caused particular ICMP message). ICMP messages of interest for MPLS are Time Exceeded and Need Fragment.

MPLS label carries not only label value, but also TTL field. When imposing label on IP packet, MPLS TTL is set to value in IP header, when last label is removed from IP packet, IP TTL is set to value in MPLS TTL. Therefore MPLS switching network can be diagnosed by means of traceroute tool that supports MPLS extension.

For example, traceroute from R5 to R1 looks like this:

```
[admin@R5] > /tool traceroute 9.9.9.1 src-address=9.9.9.5
     ADDRESS                                    STATUS
   1         4.4.4.3 15ms 5ms 5ms
```

```
                      mpls-label=17
   2          2.2.2.2 5ms 3ms 6ms
                      mpls-label=17
   3          9.9.9.1 6ms 3ms 3ms
```

Traceroute results show MPLS labels on packet when it produced ICMP Time Exceeded. The above means: when R3 received packet with MPLS TTL 1, it had label 17 on. This matches label advertised by R3 for 9.9.9.1/32. The same way R2 observed label 17 on packet on next traceroute iteration - R3 switched label 17 to label 17, as explaned above. R1 received packet without labels - R2 did penultimate hop popping as explaned above.

Drawbacks of using traceroute in MPLS network

Label switching ICMP errors

One of drawbacks of using traceroute in MPLS networks is the way MPLS handles produced ICMP errors. In IP networks ICMP errors are simply routed back to source of packet that caused the error. In MPLS network it is possible that router that produces error message does not even have route to source of IP packet (for example in case of assymetric label switching paths or some kind of MPLS tunneling, e.g. to transport MPLS VPN traffic).
Due to this produced ICMP errors are not routed to the source of packet that caused the error, but switched further along label switching path, assuming that when label switching path endpoint will receive ICMP error, it will know how to properly route it back to source.
This causes the situation, that traceroute in MPLS network can not be used the same way as in IP network - to determine failure point in the network. If label switched path is broken anywhere in the middle, no ICMP replies will come back, because they will not make it to the far endpoint of label switching path.

Penultimate hop popping and traceroute source address

Thorough understanding of pen ultimate hop behaviour and routing is necessary to understand and avoid problems that penultimate hop popping causes to traceroute.
In the example setup, regular traceroute from R5 to R1 would yield the following results:

```
[admin@R5] > /tool traceroute 9.9.9.1
     ADDRESS                                    STATUS
   1          0.0.0.0 timeout timeout timeout
   2          2.2.2.2 37ms 4ms 4ms
                      mpls-label=17
   3          9.9.9.1 4ms 2ms 11ms
```

compared to:

```
[admin@R5] > /tool traceroute 9.9.9.1 src-address=9.9.9.5
     ADDRESS                                    STATUS
   1          4.4.4.3 15ms 5ms 5ms
                      mpls-label=17
   2          2.2.2.2 5ms 3ms 6ms
                      mpls-label=17
   3          9.9.9.1 6ms 3ms 3ms
```

The reason why first traceroute does not get response from R3 is that by default traceroute on R5 uses source address 4.4.4.5 for its probes, because it is preferred source for route over which next hop to 9.9.9.1/32 is reachable:

```
[admin@R5] > /ip route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        G GATEWAY
DISTANCE             INTERFACE
 ...
 3 ADC  4.4.4.0/24         4.4.4.5                           0
ether1
 ...
 5 ADo  9.9.9.1/32                         r 4.4.4.3         110
ether1
 ...
```

When first traceroute probe is transmitted (source: 4.4.4.5, destination 9.9.9.1), R3 drops it and produces ICMP error message (source 4.4.4.3 destination 4.4.4.5) that is switched all the way to R1. R1 then sends ICMP error back - it gets switched along label switching path to 4.4.4.5.
R2 is penultimate hop popping router for network 4.4.4.0/24, because 4.4.4.0/24 is directly connected to R3. Therefore R2 removes last label and sends ICMP error to R3 unlabelled:

```
[admin@R2] > /mpls forwarding-table print
 # IN-LABEL             OUT-LABELS           DESTINATION
INTERFACE            NEXTHOP
 ...
 3 19                                      4.4.4.0/24
ether2               2.2.2.3
 ...
```

R3 drops received IP packet, because it receives packet with its own address as source address. ICMP errors produced by following probes come back correctly, because R3 receives unlabelled packets with source addresses 2.2.2.2 and 9.9.9.1, which are acceptable to route.
Command:

```
[admin@R5] > /tool traceroute 9.9.9.1 src-address=9.9.9.5
 ...
```

produces expected results, because source address of traceroute probes is 9.9.9.5. When ICMP errors are travelling back from R1 to R5, penultimate hop popping for 9.9.9.5/32 network happens at R3, therefore it never gets to route packet with its own address as source address.

Configuring VPLS

Configuring VPLS interfaces

VPLS interface can be considered tunnel interface just like EoIP interface. To achieve transparent ethernet segment forwarding between customer sites the following tunnels need to be established:

- R1-R5 (customer A)
- R1-R4 (customer A)
- R4-R5 (customer A)
- R1-R5 (customer B)

Note that each tunnel setup involves creating VPLS interfaces on both endpoints of tunnel. Negotiation of VPLS tunnels is done by LDP protocol - both endpoints of tunnel exchange labels they are going to use for tunnel. Data forwarding in tunnel then happens by imposing 2 labels on packets: tunnel label and transport label - label that ensures traffic delivery to the other endpoint of tunnel.
VPLS tunnels are configured in /interface vpls menu. vpls-id parameter identifies VPLS tunnel and must be unique for every tunnel between this and remote peer.
The necessary setup:

- on R1:

```
/interface vpls add name=A1toA2 remote-peer=9.9.9.5 mac-
address=00:00:00:00:00:a1 vpls-id=10 disabled=no
/interface vpls add name=A1toA3 remote-peer=9.9.9.4 mac-
address=00:00:00:00:00:a1 vpls-id=10 disabled=no
/interface vpls add name=B1toB2 remote-peer=9.9.9.5 mac-
address=00:00:00:00:00:b1 vpls-id=11 disabled=no
```

- on R4:

```
/interface vpls add name=A3toA1 remote-peer=9.9.9.1 mac-
address=00:00:00:00:00:a3 vpls-id=10 disabled=no
/interface vpls add name=A3toA2 remote-peer=9.9.9.5 mac-
address=00:00:00:00:00:a3 vpls-id=10 disabled=no
```

- on R5:

```
/interface vpls add name=A2toA1 remote-peer=9.9.9.1 mac-
address=00:00:00:00:00:a2 vpls-id=10 disabled=no
/interface vpls add name=A2toA3 remote-peer=9.9.9.4 mac-
address=00:00:00:00:00:a2 vpls-id=10 disabled=no
/interface vpls add name=B2toB1 remote-peer=9.9.9.1 mac-
address=00:00:00:00:00:b2 vpls-id=11 disabled=no
```

Configuring VPLS tunnel causes dynamic LDP neighbor to be created and "targeted" LDP session to be established. Targeted LDP session is session that is established between two routers that are not direct neighbors. After this setup R1 LDP neighbors are:

```
[admin@R1] > mpls ldp neighbor pr
Flags: X - disabled, D - dynamic, O - operational, T - sending-
targeted-hello, V - vpls
```

```
 #      TRANSPORT       LOCAL-TRANSPORT PEER
SEND-TARGETED ADDRESSES
 0 DO   9.9.9.2         9.9.9.1         9.9.9.2:0
no            1.1.1.2

2.2.2.2

9.9.9.2
 1 DOTV 9.9.9.5         9.9.9.1         9.9.9.5:0
yes           4.4.4.5

5.5.5.5

9.9.9.5
 2 DOTV 9.9.9.4         9.9.9.1         9.9.9.4:0
yes           3.3.3.4

5.5.5.4

9.9.9.4
```

Note that labels for IP routes are also exchanged between VPLS peers, although there is no chance any of them will be used. For example, without adding additional links, R4 will not become next hop for any route on R1, so labels learned from R4 are not likely to be ever used. Still routers maintain all labels exchanged so that they are ready for use immediately if needed. This default behaviour can be overridden by filtering which is discussed later.
By monitoring state of VPLS interface its related information can be viewed:

```
[admin@R1] /interface vpls> monitor A1toA3 once
    remote-label: 24
     local-label: 27
   remote-status:
      igp-prefix: 9.9.9.4/32
     igp-nexthop: 1.1.1.2
  imposed-labels: 21,24
```

Here we can see that R1 has assigned label 27 for tunnel between R1 and R4. MPLS forwarding table shows that this label is recognized and instead of fowarding to some next hop, received over this tunnel:

```
[admin@R1] > /mpls forwarding-table print
 # IN-LABEL             OUT-LABELS           DESTINATION
INTERFACE          NEXTHOP
 ...
11 27                                        A1toA3
 ...
```

In turn remote endpoint (R4) has assigned label 24.
igp-prefix shows route that is used to get to remote endpoint of tunnel. This implies that when forwarding traffic to remote endpoint of tunnel this router will impose transport label - label

distributed by next hop (which is shown as igp-nexthop) to 9.9.9.4/32 for 9.9.9.4/32 route. This can be confirmed on R2:

```
[admin@R2] > /mpls forwarding-table print
 # IN-LABEL             OUT-LABELS           DESTINATION
INTERFACE          NEXTHOP
 ...
 5 21                   18                   9.9.9.4/32
ether2             2.2.2.3
 ...
```

Tunnel label imposed on packets will be as assigned by remote router (R4) for this tunnel. imposed-labels reflect this setup: packets produced by tunnel will have 2 labels on them: 21 and 24.

Penultimate hop popping effects on VPLS tunnels

Penultimate hop popping of transport label causes packets to arrive at VPLS tunnel endpoint with just one tag - tunnel tag. This makes VPLS tunnel endpoint do just one label lookup to find out what to do with packet. Transport label behaviour can be observed by traceroute tool between tunnel endpoints. For example, traceroute from R1 to R4 looks like this:

```
[admin@R1] > /tool traceroute 9.9.9.4 src-address=9.9.9.1
     ADDRESS                                    STATUS
   1         1.1.1.2 7ms 5ms 3ms
                      mpls-label=21
   2         2.2.2.3 5ms 4ms 18ms
                      mpls-label=18
   3         9.9.9.4 4ms 5ms 3ms
```

traceroute output shows, that endpoint of tunnel is receiving probe without label. The same happens with VPLS tunnel traffic - at R3 transport label (18) is popped and packet is switched with just tunnel label on.
The requirement to deliver packet with tunnel label to endpoint of tunnel explains configuration advice to use "loopback" IP addresses as tunnel endpoints. If in this case R4 was establishing LDP sessions with its address 3.3.3.4, penultimate hop popping would happen not at R3, but at R2, because R3 has network 3.3.3.0/24 as its connected network (and therefore advertises implicit null label). This would cause R3 (and not R4) to receive packet with just tunnel label on, yielding unpredicted results - either dropping frame if R3 does not recognize the packet or forwarding it the wrong way.
Another issue is having VPLS tunnel endpoints directly connected, as in case of R4 and R5. There are no transport labels they can use between themselves, because they both instruct other one to be penultimate hop popping router for their tunnel endpoint address. For example on R5:

```
[admin@R5] > /mpls remote-bindings print
Flags: X - disabled, A - active, D - dynamic
 #    DST-ADDRESS        NEXTHOP         LABEL      PEER
 ...
 3 AD 9.9.9.4/32         5.5.5.4         impl-null  9.9.9.4:0
 ...
```

This causes VPLS tunnel to use only tunnel label when sending packets:

```
[admin@R5] > /int vpls monitor A2toA3 once
    remote-label: 23
     local-label: 27
   remote-status:
      igp-prefix: 9.9.9.4/32
     igp-nexthop: 5.5.5.4
  imposed-labels: 23
```

Bridging ethernet segments with VPLS

VPLS tunnels provide virtual ethernet link between routers. To transparrently connect two physical ethernet segments, they must be bridged with VPLS tunnel. In general it gets done the same way as with EoIP interfaces.
So to transparently bridge customer B networks connected to R1 and R5 the following commands are used on R1:

```
/interface bridge add name=B
/interface bridge port add bridge=B interface=ether1
/interface bridge port add bridge=B interface=B1toB2
```

and on R5:

```
/interface bridge add name=B
/interface bridge port add bridge=B interface=ether3
/interface bridge port add bridge=B interface=B2toB1
```

Note that there is not need to run (R)STP protocol on bridge as there are links between segments B1 and B2 except single VPLS tunnel between R1 and R5.

Split horizon bridging

In the example setup there are 3 tunnels set up to connect segments A1, A2 and A3, establishing so called "full mesh" of tunnels between involved segments. If bridging without (R)STP was enabled, traffic loop would occur. There are a few solutions to this:

- enabling (R)STP to eliminate the loop. This approach has a drawback - (R)STP protocol would disable forwarding through one of tunnels and keep it just for backup purposes. That way traffic between 2 segments would have to go through 2 tunnels, making setup inefficent
- using bridge firewall to make sure that traffic does not get looped - involves firewall rule setup making bridging less efficent
- using bridge horizon feature

The basic idea of split horizon bridging is to make traffic arriving over some port never be sent out some set of ports. For VPLS purposes this would mean never sending packet that arrived over one VPLS tunnel over to other VPLS tunnel, as it is known in advance, that sender of packet has connection to target network itself.
For example, if device in A1 sent packet to broadcast or unknown MAC address (which causes bridges to flood all interfaces), it would get sent to both, R5 and R4 over VPLS tunnels. In regular setup e.g. R5 when receiving such packet over VPLS tunnel would send it in A2 connected to it and also over VPLS tunnel to R4. This way R4 would get 2 copies of the same packet and further cause traffic to loop.

Bridge horizon feature allows to configure bridge ports with horizon setting so that packet received over port with horizon value X is not forwarded or flooded to any port with the same horizon value X. So in case of full mesh of VPLS tunnels, each router must be configured with the same horizon value for VPLS tunnels that are bridged together.
For example, configuration commands for R1 to enable bridging for customer A are:

```
/interface bridge add name=A
/interface bridge port add bridge=A interface=A1toA2 horizon=1
/interface bridge port add bridge=A interface=A1toA3 horizon=1
```

In similar way bridge should be configured on R4 and R5. Note that physical ethernet port is not configured with horizon value. If it was, that would disabled bridge forwarding data at all.
Note that horizon value has meaning only locally - it does not get transmitted over network, therefore it does not matter if the same value is configured in all routers participating in bridged network.

Optimizing label distribution

Label binding filtering

During implementation of given example setup, it has become clear that not all label bindings are necessary. For example, there is no need to exchange IP route label bindings between R1 and R5 or R1 and R4, as there is no chance they will ever be used. Also, if given network core is providing connectivity only for mentioned customer ethernet segments, there is no real use to distribute labels for networks that connect routers between themselves, the only routes that matter are /32 routes to endpoints of VPLS tunnels.
Label binding filtering can be used to distribute only specified sets of labels to reduce resource usage and network load.
There are 2 types of label binding filters:

- which label bindings should be advertised to LDP neighbors, configured in /mpls ldp advertise-filter
- which label bindings should be accepted from LDP neighbors, configured in /mpls ldp accept-filter

Filters are organized in ordered list, specifying prefix that must include the prefix that is tested against filter and neighbor (or wildcard).
In given example setup all routers can be configured so that they advertise labels only for routes that allow to reach endpoints of tunnels. For this 2 advertise filters need to be configured on all routers:

```
/mpls ldp advertise-filter add prefix=9.9.9.0/24 advertise=yes
/mpls ldp advertise-filter add prefix=0.0.0.0/0 advertise=no
```

This filter causes routers to advertise only bindings for routes that are included by 9.9.9.0/24 prefix which covers tunnel endpoints (9.9.9.1/32, 9.9.9.4/32, 9.9.9.5/32). The second rule is necessary because default filter result, when no rule matches is to allow action in question.
In given setup there is no need to set up accept filter, because by convention introduced by 2 abovementioned rules no LDP router will distribute unnecessary bindings.
Note that filter changes do not affect existing mappings, so to take filter into effect, connections between neighbors need to be reset. This can get done by removing them:

```
[admin@R1] /mpls ldp neighbor> print
```

```
Flags: X - disabled, D - dynamic, O - operational, T - sending-
targeted-hello, V - vpls
 #      TRANSPORT       LOCAL-TRANSPORT PEER
SEND-TARGETED ADDRESSES
 0 DO   9.9.9.2         9.9.9.1         9.9.9.2:0
no            1.1.1.2

2.2.2.2

9.9.9.2
 1 DOTV 9.9.9.5         9.9.9.1         9.9.9.5:0
yes           4.4.4.5

5.5.5.5

9.9.9.5
 2 DOTV 9.9.9.4         9.9.9.1         9.9.9.4:0
yes           3.3.3.4

5.5.5.4

9.9.9.4
[admin@R1] /mpls ldp neighbor> remove [find]
```

So on R1, for example we get:

```
[admin@R1] > /mpls remote-bindings print
Flags: X - disabled, A - active, D - dynamic
 #    DST-ADDRESS        NEXTHOP         LABEL      PEER
 0  D 9.9.9.1/32                         30         9.9.9.5:0
 1  D 9.9.9.5/32                         impl-null  9.9.9.5:0
 2  D 9.9.9.4/32                         31         9.9.9.5:0
 3  D 9.9.9.2/32                         32         9.9.9.5:0
 4  D 9.9.9.3/32                         33         9.9.9.5:0
 5 AD 9.9.9.2/32         1.1.1.2         impl-null  9.9.9.2:0
 6  D 9.9.9.1/32                         24         9.9.9.2:0
 7 AD 9.9.9.3/32         1.1.1.2         25         9.9.9.2:0
 8 AD 9.9.9.4/32         1.1.1.2         26         9.9.9.2:0
 9 AD 9.9.9.5/32         1.1.1.2         27         9.9.9.2:0
10  D 9.9.9.1/32                         27         9.9.9.4:0
11  D 9.9.9.5/32                         28         9.9.9.4:0
12  D 9.9.9.4/32                         impl-null  9.9.9.4:0
13  D 9.9.9.2/32                         29         9.9.9.4:0
14  D 9.9.9.3/32                         30         9.9.9.4:0
```

There still are unnecessary bindings, this time - the bindings distributed due to establishing targeted LDP session with remote endpoints of VPLS tunnels (bindings from 9.9.9.5 and 9.9.9.4) To filter out those, we configure routers to not distribute any IP bindings to any of tunnel endpoint routers. For example on R1, filter table should look like this:

```
[admin@R1] /mpls ldp advertise-filter> print
Flags: X - disabled
 #   PREFIX             NEIGHBOR        ADVERTISE
 0   0.0.0.0/0          9.9.9.4         no
 1   0.0.0.0/0          9.9.9.5         no
 2   9.9.9.0/24         all             yes
 3   0.0.0.0/0          all             no
```

This causes routers to have minimal label binding tables, for example on R1:

```
[admin@R1] > /mpls local-bindings print
Flags: X - disabled, A - advertised, D - dynamic, L - local-
route, G - gateway-route, e - egress
 #      DST-ADDRESS        LABEL      PEERS
 0 ADLe 9.9.9.1/32         impl-null  9.9.9.2:0
 1 ADG  9.9.9.3/32         40         9.9.9.2:0
 2 ADG  9.9.9.4/32         41         9.9.9.2:0
 3 ADG  9.9.9.2/32         42         9.9.9.2:0
 4 ADG  9.9.9.5/32         43         9.9.9.2:0
[admin@R1] > /mpls remote-bindings print
Flags: X - disabled, A - active, D - dynamic
 #    DST-ADDRESS        NEXTHOP         LABEL      PEER
 0 AD 9.9.9.2/32         1.1.1.2         impl-null  9.9.9.2:0
 1  D 9.9.9.1/32                         24         9.9.9.2:0
 2 AD 9.9.9.3/32         1.1.1.2         25         9.9.9.2:0
 3 AD 9.9.9.4/32         1.1.1.2         26         9.9.9.2:0
 4 AD 9.9.9.5/32         1.1.1.2         27         9.9.9.2:0
```

Note that IP binding distribution should not be disabled between R4 and R5 although they are tunnel endpoints. Doing so would not harm regular case, because R4 and R5 does not need IP bindings to VPLS tunnel data, but in case link between R3 and R5 would go down, all traffic to R5 from R1 would have to be rerouted through R4. In such case R5 not distributing IP bindings to R4 would cause R4 to not be able to forward MPLS traffic to R5.

Effects of label binding filtering on data forwarding in network

Note the traceroute results after these changes. Traceroute from R1 to R5 using R1 loopback address as source address still behaves the same - each hop reports received labels:

```
[admin@R1] > tool traceroute 9.9.9.5 src-address=9.9.9.1
     ADDRESS                                    STATUS
   1         1.1.1.2 11ms 4ms 5ms
                      mpls-label=27
   2         2.2.2.3 4ms 4ms 4ms
                      mpls-label=25
   3         9.9.9.5 12ms 3ms 3ms
```

But in case of traceroute using R1 address that faces network:

```
[admin@R1] > tool traceroute 9.9.9.5 src-address=1.1.1.1
     ADDRESS                                    STATUS
   1         0.0.0.0 timeout timeout timeout
```

```
   2          0.0.0.0 timeout timeout timeout
   3          9.9.9.5 3ms 3ms 3ms
```

Now all hops except the last one do not respond. The reason for this is the fact, that there is no label switching path back from R5 to R1 (which this time uses address 1.1.1.1) because there are no label bindings distributed - so ICMP response is routed and routers on the way back (R3 and R2) receive packets with their own source address and drop them right away without routing.
On the other hand, traceroute from R1 to R5 using their non-loopback addresses:

```
[admin@R1] > tool traceroute 4.4.4.5 src-address=1.1.1.1
     ADDRESS                                    STATUS
   1         1.1.1.2 13ms 1ms 1ms
   2         2.2.2.3 3ms 2ms 2ms
   3         4.4.4.5 3ms 3ms 23ms
```

There is no label switching involved doing this traceroute and it works just like in network without MPLS at all.

EXP bit behaviour

MPLS label EXP field overview

When MPLS label is attached to packet, it increases packet length by 32 bits (4 bytes). These 32 bits are broken down as follows:

- label value itself (20 bits)
- EXP ("experimental") field (3 bits)
- time to live field (8 bits)
- bottom of stack field (1 bit)

Use of "experimental" bits is not specified by MPLS standards, but most common use is to carry QoS information, similar to 802.1q priority in VLAN tag. Note that EXP field is 3 bits only therefore it can carry values from 0 to 7 only, which allows to have 8 traffic classes.

EXP field treatment in HauteRouterOS

When HauteRouterOS receives MPLS packet, it sets "ingress priority" value for packet to that carried inside top label. Note that "ingress priority" **is not** a field inside packet headers - it can be thought of like additional mark assigned to packet while being processed by router. When HauteRouterOS labels MPLS packet, it sets EXP bits to "priority" (not "ingress priority"!) assigned to packet. When HauteRouterOS switches MPLS packet, "ingress priority" is automatically copied to "priority", this way regular MPLS switching communicates priority info over whole label switched path.

Additional info on "ingress priority" and "priority" handling is also in WMM.

Therefore what happens to EXP field depends based on what action is taken on packet:

- if packet is MPLS switched (by popping label off packet and pushing on new one), EXP field in new label will be the same as in received label, because:
  - RouterOS sets "ingress priority" to EXP bits in received label
  - Switching automatically sets "priority" to "ingress priority"
  - RouterOS labels packet with new label and sets its EXP bits to value in "priority".
- if packet is MPLS switched by using penultimate-hop-popping (received label is popped off and no new one is pushed on), EXP field of received priority stays in "priority" field of packet and may be used by some other MAC protocol, e.g. WMM or 802.1q VLAN, for example:
  - RouterOS sets "ingress priority" to EXP bits in received label
  - Switching automatically sets "priority" to "ingress priority"
  - RouterOS switches packet to next hop (without pushing on label) and that happens over VLAN interface
  - VLAN interface sets 802.1q priority in VLAN header to "priority" value of packet.

Note that penultimate-hop-popping can therefore loose QoS information carried over label switched path at the last hop. In cases where this is not desirable, penultimate-hop-popping behaviour should be disabled by using Explicit NULL label instead of Implicit NULL label for last hop in label switched path. Using Explicit NULL label for last hop is default behaviour for MPLS TE tunnels.

- if packet is supposed to be sent over label switched path (first label will get pushed on packet), EXP bits will be set to value in "priority", which in turn can be set up properly using firewall rules or other means (e.g. from DSCP field in IP header)
- if packet is received for local processing, "ingress priority" is set to EXP field of received packet and can therefore be used to update DSCP field of packet or set "priority" from "ingress priority" using firewall rules

MPLS RSVP TE

MPLS RSVP TE tunnels are a way to establish unidirectional label switching paths. In general RSVP TE serves similar purpose as label distribution using LDP protocol - establishing label switched path that ensures frame delivery from ingress to egress router, but with additional features:

- possibility to establish label switching path using either full or partial explicit route;

- constraint based LSP establishment - label switching path is established over links that fulfill requirements, such as bandwidth and link properties.

MPLS RSVP TE is based on RSVP protocol with extensions introduced by RFC 3209 that adds support for explicit route and label exchange.

Note that constraints for path establishment are purely controlled by administrator - for example, bandwidth of link participating in RSVP TE network is set by administrator and does not necessarily reflect real bandwidth of the link. The same way bandwidth reserved for tunnel is set by administrator and does not automatically imply any limits on traffic sent over tunnel. Therefore at any moment in time, bandwidth available on TE link is bandwidth configured for link minus sum of all reservations made on link, not physically available bandwidth which can be either less (in case data is forwarded over tunnels with rate that exceeds bandwidth reserved for tunnel or if non-RSVP tunnel data is forwarded over link as well) or more (in case data is forwarded over tunnels with rate smaller than allocated for tunnel) than bandwidth available for reservations.

RSVP TE tunnels are initiated by head-end (ingress router) of tunnel. Head-end router sends RSVP Path message containing necessary parameters towards tail-end of the tunnel. Routers along the path ensure that they can forward Path message towards next hop, taking into acount path constraints. Once Path message reaches tail-end of the tunnel, tail-end router sends RSVP Resv message in the opposite direction. Resv message hop by hop traverses exactly the same path that Path message, only in the opposite direction. Each router forwarding Resv message allocates necessary bandwith on appropriate downstream link if possible. Once head-end router succesfully receives Resv message that matches sent Path message, tunnel can be considered established. Tunnel is maintained by periodically refreshing its state using Path and Resv messages.

RSVP TE tunnels can be established with number of path options:

- along path that data from head-end of tunnel is routed to tail-end - in this case each router along tunnel path figures out next hop of tunnel based on routing table. If at some point usable route is not found or downstream interface does not meet constraints (for example if requested bandwidth exceeds available bandwidth), tunnel can not be established.
- along statically configured explicit path - in this case each router along tunnel path figures out next hop of tunnel based on explicit route specified in Path message. This explicit route can be either complete (specifies all routers along the path in the order they must be traversed) or partial (specifies only some routers that must be traversed). To decide next hop router, each router along the path look up route to next router specified in explicit route. If no usable route is found or downstream interface does not meet constraints, tunnel can not be established
- Constrained Shortest Path First - in this case head-end router calculates path to tail-end using its knowledge of network state - properties of links and available

bandwidth. This option needs assistance from IGP routing protocol (such as OSPF) to distribute bandwidth information throughout the network. This is implemented in OSPF by means of opaque LSAs. When using CSPF, head-end router calculates path that satisfies the requirements and produces explicit path for Path message. If path that matches constraints can not be calculated, tunnel can not be established. Dynamically calculated path can also be partially explicit - in this case CSPF seeks for shortest path matching constraints between every two explicit hops. If explicit path is specified completely and CSPF is used, CSPF just checks if this path meets the constraints taking into account knowledge about link states in network - so instead of failure to establish tunnel while forwarding Path message in network, Path message is not even sent as it is clear that establishing tunnel will fail.

Forwarding traffic onto TE tunnels

RSVP TE tunnel head-end appears as interface in RouterOS. Note that RSVP TE tunnels are unidirectional - it is not necessary to have matching tunnel for reverse direction on tail-end router. When tail-end router receives data sent over tunnel, it either receives it with TE tunnel label stripped off by penultimate hop (non-default behaviour) or with explicit-null label, which gets stripped and packet is further inspected (if tunnel label is last label in stack, packet gets routed, otherwise it is processed based on next label in stack, for example, as VPLS packet). Bidirectional tunnel can be simulated by creating one tunnel in one direction and other in other direction between the same endpoints. Still no data will be accounted as received over TE tunnel, as in reality both tunnels are unrelated.

One way to forward traffic onto tunnel is to use routing, but this limits TE tunnel to be used only for routing IP packets.

Additionally, several types of traffic can be forwarded onto TE tunnel automatically, if it is known to be destined to the endpoint of tunnel and if tunnel is active:

- traffic that is routed using route route learned from BGP, if BGP NextHop is tunnel endpoint (this default behaviour can be changed by setting route porperty "use-te-nexthop" to "no"), both - regular IP and VPNv4 (MP-BGP IP VPN) routes fit in this category;
- traffic for VPLS interfaces, if remote endpoint of VPLS pseudowire is the same as TE tunnel endpoint.

For example, for IP BGP route having BGP NextHop x.x.x.x, forwarding method will be chosen according to the following rules:

- if TE tunnel with endpoint x.x.x.x is active, use it;
- otherwise if LDP label mapping from next hop towards x.x.x.x is received, use it;

- otherwise use regular routing (no MPLS encapsulation).

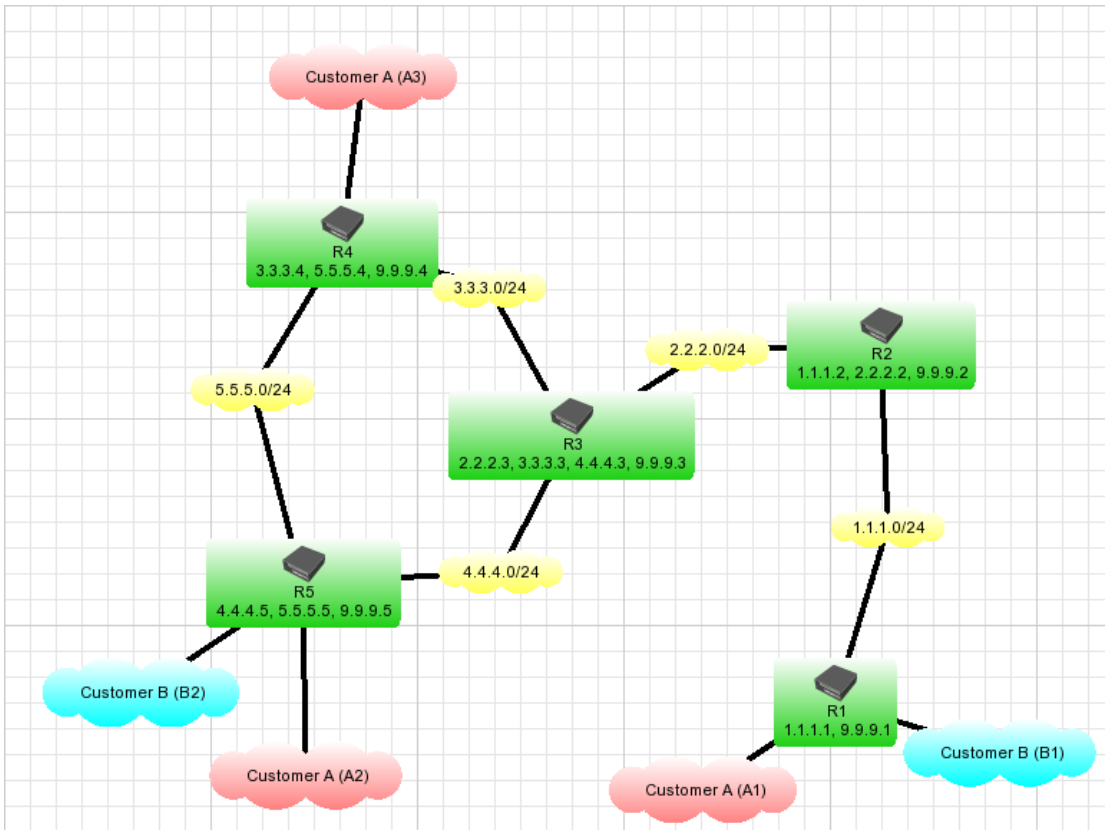
In similar way, if remote address of VPLS pseudowire is x.x.x.x, forwarding method will be chosen in the following order:

- if TE tunnel with endpoint x.x.x.x is active, use it;
- otherwise if LDP label mapping from next hop towards x.x.x.x is received, use it;
- otherwise VPLS tunnel can not be active.

Note that RSVP TE tunnels as a way to establish LSPs can be used together with LDP. Using RSVP TE does not replace or disable LDP, but LSP established by TE is usually preferred over one established using LDP.

Example network

Consider the same network as used for LDP signaled VPLS example in MPLS VPLS:



Customer A wants to establish IP VPN between his 3 sites and Customer B wants to transparent connection for ethernet segments at his sites.

Prerequisites for MPLS TE

In general, prerequisites for using MPLS TE are the same as mentioned in MPLSVPLS, but there are a few details:

- by default TE tunnel tail-end router advertises explicit null label, therefore penultimate hop popping does not happen (the purpose of using explicit null label is to communicate QoS information in MPLS label Exp field), so main purpose of having "loopback" IP address for every router is to have tunnel endpoints unaffected by link state changes;
- in order to use CSPF path selection for tunnels, OSPF must be configured and running in network.

Enabling TE support

In order for OSPF to distribute TE information, TE related OSPF parameters must be set:

```
[admin@R1] > /routing ospf set mpls-te-area=backbone mpls-te-router-
id=lobridge
```

This instructs OSPF to distribute TE information in "backbone" area using IP address of "lobridge" as router ID.

In order for router to be able to participate in TE tunnel (either as head-end, tail-end or forwarding router), TE support must be enabled. TE support must be enabled on all interfaces that will receive and send RSVP TE protocol packets. On R1 it is done by commands (interface ether3 is facing network 1.1.1.0/24):

```
[admin@R1] > /mpls traffic-eng interface add interface=ether3
bandwidth=100000
```

This configures ether3 interface with TE support, having bandwidth 100000 Bps. Other routers are configured in similar way.

As soon as TE support is enabled on interface, appropriate opaque LSAs are distributed into OSPF area. For example, on R1 it can be seen, that there is total 15 opaque LSAs in LSA database:

```
[admin@R1] > /routing ospf lsa print
  ...
backbone
opaque-area  1.0.0.0        1.1.1.2        0x80000004      1038
backbone
opaque-area  1.0.0.0        2.2.2.3        0x80000004      1039
```

```
backbone
opaque-area  1.0.0.0        3.3.3.4        0x80000004      1038
backbone
opaque-area  1.0.0.0        4.4.4.5        0x80000004      1038
backbone
opaque-area  1.0.0.0        11.11.11.1     0x80000004      1037
backbone
opaque-area  1.0.0.1        1.1.1.2        0x80000004      1038
backbone
opaque-area  1.0.0.1        2.2.2.3        0x80000004      1039
backbone
opaque-area  1.0.0.1        3.3.3.4        0x80000004      1037
backbone
opaque-area  1.0.0.1        4.4.4.5        0x80000004      1038
backbone
opaque-area  1.0.0.2        1.1.1.2        0x80000004      1038
backbone
opaque-area  1.0.0.2        2.2.2.3        0x80000004      1039
backbone
opaque-area  1.0.0.2        3.3.3.4        0x80000004      1037
backbone
opaque-area  1.0.0.2        4.4.4.5        0x80000004      1038
backbone
opaque-area  1.0.0.3        2.2.2.3        0x80000004      1039
backbone
opaque-area  1.0.0.3        11.11.11.1     0x80000004      1037
  ...
```

Creating basic TE tunnel

Assume that we want to create TE tunnel from R1 to R5. In order to do this, tunnel path specification must be created:

```
[admin@R1] > /mpls traffic-eng tunnel-path add use-cspf=yes name=dyn
```

This creates path template for purely dynamic path that will use CSPF.

Next, TE tunnel itself must be created:

```
[admin@R1] /interface traffic-eng> add name=te1 bandwidth=1000 primary-
path=dyn from-address=9.9.9.1 to-address=9.9.9.5 disabled=no record-
route=yes
```

We can monitor tunnel to see its state:

```
[admin@R1] /interface traffic-eng> monitor 0
             tunnel-id: 7
    primary-path-state: established
          primary-path: dyn
  secondary-path-state: not-necessary
           active-path: dyn
```

```
          active-lspid: 1
          active-label: 29
        explicit-route:
"S:1.1.1.2/32,S:2.2.2.2/32,S:2.2.2.3/32,S:4.4.4.3/32,S:4.4.4.5/32"
        recorded-route: "1.1.1.2[30],2.2.2.3[29],4.4.4.5[0]"
```

Notice, that CSPF has created explicit route that traverses R2, R3 and R5 (tail-end). TE tunnel was requested to record route it is traversing (by "record-route=yes" setting), recorded route is displayed in status along with labels that particular router has allocated for this tunnel.

Once TE tunnel is established, VPLS interface from R1 to R5 automatically switches to use this TE tunnel:

```
[admin@R1] /interface vpls> monitor 0
       remote-label: 24
        local-label: 25
      remote-status:
          transport: te1
  transport-nexthop: 1.1.1.2
     imposed-labels: 30,24
```

On routers in between R1 and R5, RSVP path and reservation state can be monitored, for example on R2:

```
[admin@R2] > /mpls traffic-eng path-state print
Flags: L - locally-originated, E - egress, F - forwarding, P - sending-
path, R - sending-resv
 #      SRC                  DST                  BANDWIDTH  OUT-
INTERFACE                                          OUT-NEXT-HOP
 0  FPR 9.9.9.1:1            9.9.9.5:2            1000       ether2
2.2.2.3
[admin@R2] > /mpls traffic-eng resv-state print
Flags: E - egress, A - active, N - non-output, S - shared
 #    SRC                  DST                  BANDWIDTH  LABEL
INTERFACE                   NEXT-HOP
 0 AS 9.9.9.1:1            9.9.9.5:7            1000       30
ether2                      2.2.2.3
```

Note, that available bandwidth on ether2 interface (connected to R3) on R2 has changed:

```
[admin@R2] > /mpls traffic-eng interface print
Flags: X - disabled, I - invalid
 #   INTERFACE
BANDWIDTH  TE-METRIC  REMAINING-BW
 0   ether1
100000     1          100000
 1   ether2
100000     1          99000
```

AutoBandwidth Control

By default MPLS TE tunnels do not apply any rate limitation on traffic that gets sent over tunnel. That way "bandwidth" settings for MPLS TE enabled interfaces and TE tunnels are only used for reservation accounting. There are also no means to adjust bandwidth that gets reserved for tunnel other than changing tunnel configuration no matter what is actual amount of traffic sent over tunnel. To make TE tunnels more flexible and easy to use, the following features have been introduced:

- Bandwidth limitation
- Automatic bandwidth adjustment

These features operate on tunnel head end (ingress) router. These features can either be used alone or in combination.

Bandwidth limitation

TE tunnel can be configured to limit the rate at which traffic is allowed to enter the tunnel. Limit is specified on ingress router in percent of tunnel bandwidth. E.g. creating the following tunnel:

```
[admin@R1] /interface traffic-eng> add name=te1 from-address=9.9.9.1
to-address=9.9.9.5 bandwidth=100000 bandwidth-limit=120
primary-path=stat
```

means that tunnel will reserve bandwidth of 100 kilobits per second across MPLS backbone from 9.9.9.1 to 9.9.9.5 and that ingress router will limit the rate of traffic entering the tunnel to 120 kilobits per second (120% of 100 kilobits per second bandwidth). This can be confirmed by monitoring tunnel interface:

```
[admin@R1] /interface traffic-eng> monitor te1
              tunnel-id: 3
     primary-path-state: established
           primary-path: stat
   secondary-path-state: not-necessary
            active-path: stat
           active-lspid: 1
           active-label: 20
     reserved-bandwidth: 100.0kbps
             rate-limit: 120.0kbps
     rate-measured-last: 0bps
  rate-measured-highest: 0bps
```

Note that by default any limiting is disabled. By specifying limit as percentage of tunnel bandwidth, TE tunnel bandwith limits can be configured in rather flexible ways - some

tunnels can be configured to hard limit while others can be configured with reasonable reserve, achieving different classes of service.

Automatic bandwidth adjustment

Auto bandwidth adjustment feature enables MPLS TE network to follow the changes of amount of data transmitted over tunnel. Bandwidth adjustment feature works as follows:

- Actual amount of data entering tunnel during averaging interval (**auto-bandwidth-avg-interval**) is measured, producing average rate.
- Tunnel keeps track of highest average rate seen during update interval (**auto-bandwidth-update-interval**)
- When update interval expires, TE tunnel bandwidth is updated to highest observed average rate, taking into account specified range over which bandwidth is allowed to change (**auto-bandwidth-range**)

Auto bandwidth adjustment feature gets enabled by specifying **auto-bandwidth-range**. For example, adding the following tunnel:

```
[admin@R1] /interface traffic-eng> add name=te1 from-address=9.9.9.1
to-address=9.9.9.5 bandwidth=100000
primary-path=stat auto-bandwidth-range=10000-500000 auto-bandwidth-avg-
interval=10s auto-bandwidth-update-interval=1m
```

means that tunnel will measure average rate over 10 second periods and once per minute will update bandwidth in range from 10 to 500 kilobits per second. Tunnel **bandwidth** setting specifies the initial bandwidth of tunnel. The above tunnel in complete absence of data over it after 1 minute will change its bandwidth to specified minimum 10 kbps:

```
[admin@R1] /interface traffic-eng> monitor te1
              tunnel-id: 3
     primary-path-state: established
           primary-path: stat
   secondary-path-state: not-necessary
            active-path: stat
           active-lspid: 2
           active-label: 21
     reserved-bandwidth: 10.0kbps
             rate-limit: 12.0kbps
     rate-measured-last: 0bps
  rate-measured-highest: 0bps
```

Additionally, tunnel can be configured to reserve more bandwidth than measured. This can be achieved with **auto-bandwidth-reserve** setting which specifies percentage of additional bandwidth to reserve - so setting **auto-bandwith-reserve** to 10 means that tunnel will reserve 10% more bandwidth than measured (but will still obey the **auto-**

bandwidth-range). For example changing above tunnel and running constant stream of 50kbps through it will yield the following results:

```
[admin@R1] /interface traffic-eng> set te1 auto-bandwidth-reserve=30
```

In the beginning tunnel reserves its initially specified bandwidth:

```
[admin@R1] /interface traffic-eng> monitor te1
              tunnel-id: 6
     primary-path-state: established
           primary-path: stat
   secondary-path-state: not-necessary
            active-path: stat
           active-lspid: 1
           active-label: 27
     reserved-bandwidth: 100.0kbps
             rate-limit: 120.0kbps
     rate-measured-last: 48.8kbps
  rate-measured-highest: 48.8kbps
```

After update period and after previous reservations are torn down notice how reserved bandwidth exceeds average rate by 30%. Also notice that **rate-limit** correctly changes to 120% of **reserved-bandwidth**:

```
[admin@R1] /interface traffic-eng> monitor te1
              tunnel-id: 6
     primary-path-state: established
           primary-path: stat
   secondary-path-state: not-necessary
            active-path: stat
           active-lspid: 2
           active-label: 28
     reserved-bandwidth: 64.4kbps
             rate-limit: 77.3kbps
     rate-measured-last: 48.8kbps
  rate-measured-highest: 48.8kbps
```

Note that in case reservation must be updated to lower value, brief period after update period **reserved-bandwidth** will still display previous reservation value. The reason for this is that new reservation is made without disrupting the previous tunnel and therefore shares its reservation until old reservation is torn down. **rate-limit** on turn is correctly updated to intended value. In the above example, after stopping the 50kbps stream and after update period will pass with tunnel being idle, for a brief period after update tunnel info can be:

```
[admin@R1] /interface traffic-eng> monitor te1
              tunnel-id: 6
     primary-path-state: established
           primary-path: stat
   secondary-path-state: not-necessary
```

```
            active-path: stat
           active-lspid: 2
           active-label: 34
     reserved-bandwidth: 63.4kbps
             rate-limit: 12.0kbps
     rate-measured-last: 0bps
  rate-measured-highest: 0bps
```

After previous reservation (63.4kbps) is torn down, **reserved-bandwidth** correctly changes to 10kbps:

```
[admin@R1] /interface traffic-eng> monitor 1
              tunnel-id: 6
     primary-path-state: established
           primary-path: stat
   secondary-path-state: not-necessary
            active-path: stat
           active-lspid: 2
           active-label: 34
     reserved-bandwidth: 10.0kbps
             rate-limit: 12.0kbps
     rate-measured-last: 0bps
  rate-measured-highest: 0bps
```

Note that **auto-bandwidth-reserve** is applied to actual measured bandwidth, before range checking according to **auto-bandwidth-range** - therefore 10kbps gets reserved, instead of 13kbps.

Combining bandwidth limitation with automatic bandwidth adjustment

Auto bandwidth adjustment can be used in combination with bandwidth limit feature - **bandwidth-limit** setting will apply to bandwidth actually reserved for tunnel. In order to successfully cobine both features, actual bandwidth must be allowed to fluctuate to some extent - e.g. if **bandwidth-limit** will be configured to 100% (this effectively means that rate will be limited to the bandwidth reserved for tunnel), tunnel will not have any chance to increase its reservation. Therefore either **bandwidth-limit** should be configured to more than 100%, or **auto-bandwidth-reserve** should be configured to more than 0%.

Prerequisites for MPLS

"Loopback" IP address

Although not a strict requirement, it is advisable to configure routers participating in MPLS network with "loopback" IP addresses (not attached to any real network interface) to be used by LDP to establish sessions.

This serves 2 purposes:

- as there is only one LDP session between any 2 routers, no matter how many links connect them, loopback IP address ensures that LDP session is not affected by interface state or address changes
- use of loopback address as LDP transport address ensures proper penultimate hop popping behaviour when multiple labels are attached to packet as in case of VPLS

In RouterOS "loopback" IP address can be configured by creating dummy bridge interface without any ports and adding address to it. For example, on R1 it is done with the following commands:

```
/interface bridge add name=lobridge
/ip address add address=9.9.9.1/32 interface=lobridge
```

The rest of routers are configured similar way.

IP connectivity

As LDP distributes labels for active routes, essential requirement is properly configured IP routing. LDP by default distributes labels for active IGP routes (that is - connected, static, and routing protocol learned routes, except BGP).

In given example setup OSPF is used to distribute routes. For example, on R5 OSPF is configured with the following commands:

```
/routing ospf set redistribute-connected=as-type-1
/routing ospf network add area=backbone network=4.4.4.0/24
/routing ospf network add area=backbone network=5.5.5.0/24
```

On other routers OSPF is configured in similar way.

This yields routing table on R5 like this:

```
[admin@R5] > ip route print
Flags: X - disabled, A - active, D - dynamic, C - connect, S - static,
r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        GATEWAY-STATE GATEWAY
DISTANCE             INTERFACE
 0 ADo  1.1.1.0/24                         reachable     4.4.4.3
110                  ether1
```

```
 1 ADo  2.2.2.0/24                          reachable     4.4.4.3
110                  ether1
 2 ADo  3.3.3.0/24                          reachable     4.4.4.3
110                  ether1
                                            reachable     5.5.5.4
ether2
 3 ADC  4.4.4.0/24        4.4.4.5
0                    ether1
 4 ADC  5.5.5.0/24        5.5.5.5
0                    ether2
 5 ADo  9.9.9.1/32                          reachable     4.4.4.3
110                  ether1
 6 ADo  9.9.9.2/32                          reachable     4.4.4.3
110                  ether1
 7 ADo  9.9.9.3/32                          reachable     4.4.4.3
110                  ether1
 8 ADo  9.9.9.4/32                          reachable     5.5.5.4
110                  ether2
 9 ADC  9.9.9.5/32        9.9.9.5
0                    lobridge
```

and traceroute from R5 to R1 like this:

```
[admin@R5] > /tool traceroute 9.9.9.1
     ADDRESS                                    STATUS
   1         4.4.4.3 11ms 1ms 4ms
   2         2.2.2.2 23ms 3ms 2ms
   3         9.9.9.1 26ms 3ms 3ms
```

Configuring LDP

In order to distribute labels for routes, LDP should get enabled. On R1 this is done by commands (interface ether3 is facing network 1.1.1.0/24):

```
/mpls ldp set enabled=yes transport-address=9.9.9.1 lsr-id=9.9.9.1
/mpls ldp interface add interface=ether3
```

Note that transport-address gets set to 9.9.9.1. This makes router originate LDP session connections with this address and also advertise this address as transport address to LDP neighbors.

Other routers are configured in similar way - LDP is enabled on interfaces that connect routers and not enabled on interfaces that connect customer networks. For example, on R5:

```
[admin@R5] > /ip address print
Flags: X - disabled, I - invalid, D - dynamic
 #   ADDRESS            NETWORK         BROADCAST       INTERFACE
 0   4.4.4.5/24         4.4.4.0         4.4.4.255       ether1
 1   5.5.5.5/24         5.5.5.0         5.5.5.255       ether2
```

```
 2   9.9.9.5/32         9.9.9.5         9.9.9.5         lobridge
[admin@R5] > /mpls ldp interface print
Flags: I - invalid, X - disabled
 #   INTERFACE                                           HELLO-
INTERVAL       HOLD-TIME
 0   ether1                                              5s
15s
 1   ether2                                              5s
15s
```

After LDP sessions are established, on R5 there are 2 LDP neighbors:

```
[admin@R5] > /mpls ldp neighbor print
Flags: X - disabled, D - dynamic, O - operational, T - sending-
targeted-hello, V - vpls
 #      TRANSPORT       LOCAL-TRANSPORT PEER
SEND-TARGETED ADDRESSES
 0 DO   9.9.9.4         9.9.9.5         9.9.9.4:0
no             3.3.3.4

5.5.5.4

9.9.9.4
 1 DO   9.9.9.3         9.9.9.5         9.9.9.3:0
no             2.2.2.3

3.3.3.3

4.4.4.3

9.9.9.3
```

/mpls local-bindings shows labels that this router has assigned to routes and peers it has distributed the label to. It shows that R5 has distributed labels for all its routes to both of its neighbors - R3 and R4

```
[admin@R5] > /mpls local-bindings print
Flags: X - disabled, A - advertised, D - dynamic, L - local-route, G -
gateway-route, e - egress
 #      DST-ADDRESS        LABEL      PEERS
 0 ADLe 4.4.4.0/24         impl-null  9.9.9.4:0
                                      9.9.9.3:0
 1 ADLe 9.9.9.5/32         impl-null  9.9.9.4:0
                                      9.9.9.3:0
 2 ADG  9.9.9.4/32         17         9.9.9.4:0
                                      9.9.9.3:0
 3 ADLe 5.5.5.0/24         impl-null  9.9.9.4:0
                                      9.9.9.3:0
 4 ADG  1.1.1.0/24         18         9.9.9.4:0
                                      9.9.9.3:0
 5 ADG  2.2.2.0/24         19         9.9.9.4:0
                                      9.9.9.3:0
 6 ADG  9.9.9.1/32         20         9.9.9.4:0
```

```
                                        9.9.9.3:0
 7 ADG  9.9.9.2/32          21          9.9.9.4:0
                                        9.9.9.3:0
 8 ADG  9.9.9.3/32          22          9.9.9.4:0
                                        9.9.9.3:0
 9 ADG  3.3.3.0/24          23          9.9.9.4:0
                                        9.9.9.3:0
```

/mpls remote-bindings shows labels that are allocated for routes by neighboring routers and advertised to this router:

```
[admin@R5] > /mpls remote-bindings print
Flags: X - disabled, A - active, D - dynamic
 #    DST-ADDRESS        NEXTHOP         LABEL      PEER
 0  D 4.4.4.0/24                         16         9.9.9.4:0
 1 AD 3.3.3.0/24         5.5.5.4         impl-null  9.9.9.4:0
 2  D 9.9.9.5/32                         17         9.9.9.4:0
 3 AD 9.9.9.4/32         5.5.5.4         impl-null  9.9.9.4:0
 4  D 5.5.5.0/24                         impl-null  9.9.9.4:0
 5  D 1.1.1.0/24                         18         9.9.9.4:0
 6  D 2.2.2.0/24                         19         9.9.9.4:0
 7  D 9.9.9.1/32                         20         9.9.9.4:0
 8  D 9.9.9.2/32                         21         9.9.9.4:0
 9  D 9.9.9.3/32                         22         9.9.9.4:0
10 AD 1.1.1.0/24         4.4.4.3         16         9.9.9.3:0
11 AD 2.2.2.0/24         4.4.4.3         impl-null  9.9.9.3:0
12  D 4.4.4.0/24                         impl-null  9.9.9.3:0
13  D 3.3.3.0/24                         impl-null  9.9.9.3:0
14 AD 9.9.9.1/32         4.4.4.3         17         9.9.9.3:0
15 AD 9.9.9.3/32         4.4.4.3         impl-null  9.9.9.3:0
16  D 9.9.9.4/32                         18         9.9.9.3:0
17  D 5.5.5.0/24                         19         9.9.9.3:0
18 AD 9.9.9.2/32         4.4.4.3         20         9.9.9.3:0
19  D 9.9.9.5/32                         21         9.9.9.3:0
```

Here we can observe that R5 has received label bindings for all routes from both its neighbors - R3 and R4, but only the ones for whom particular neighbor is next hop are active. For example:

```
[admin@R5] > /ip route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        G GATEWAY         DISTANCE
INTERFACE
...
 5 ADo  9.9.9.1/32                         r 4.4.4.3         110
ether1
...
[admin@R5] > /mpls remote-bindings print
Flags: X - disabled, A - active, D - dynamic
 #    DST-ADDRESS        NEXTHOP         LABEL      PEER
```

```
...
 7  D 9.9.9.1/32                          20         9.9.9.4:0
...
14 AD 9.9.9.1/32         4.4.4.3         17         9.9.9.3:0
...
```

From the above we see that R3, which is next hop for network 9.9.9.1/32 from R5 perspective, has assigned label 17 for traffic going to 9.9.9.1/32. This implies that when R5 will be routing traffic to this network, will impose label 17.

Label switching rules can be seen in /mpls forwarding-table. For example, on R3 it looks like this:

```
[admin@R3] > /mpls forwarding-table print
 # IN-LABEL             OUT-LABELS           DESTINATION
INTERFACE            NEXTHOP
 ...
 2 17                   17                   9.9.9.1/32         ether1
2.2.2.2
 ...
```

This rule says that R3 receiving packet with label 17 will change it to label 17 assigned by R2 for network 9.9.9.1/32 (R2 is next hop for 9.9.9.1/32 from R3 perspective):

```
[admin@R2] > /mpls local-bindings print
Flags: X - disabled, A - advertised, D - dynamic, L - local-route, G -
gateway-route, e - egress
 #      DST-ADDRESS        LABEL      PEERS
 ...
 3 ADG  9.9.9.1/32         17         9.9.9.1:0
                                      9.9.9.3:0
 ...
```

R2 MPLS forwarding table tells:

```
[admin@R2] > /mpls forwarding-table print
 # IN-LABEL             OUT-LABELS           DESTINATION
INTERFACE            NEXTHOP
 ...
 1 17                                        9.9.9.1/32         ether1
1.1.1.1
 ...
```

Notice, that forwarding rule does not have any out-labels. The reason for this is that R2 is doing penultimate hop popping for this network. R1 does not assign any real label for 9.9.9.1/32 network, because it is known that R1 is egress point for 9.9.9.1/32 network (router is egress point for networks that are directly connected to it, because next hop for traffic is not MPLS router), therefore is advertises "implicit null" label for this route:

```
[admin@R2] > /mpls remote-bindings print
Flags: X - disabled, A - active, D - dynamic
 #    DST-ADDRESS        NEXTHOP         LABEL      PEER
 ...
13 AD 9.9.9.1/32         1.1.1.1         impl-null  9.9.9.1:0
 ...
```

This tells R2 to forward traffic for 9.9.9.1/32 to R1 unlabelled, which is exactly what R2 mpls forwarding-table entry tells. Penultimate hop popping ensures that routers do not have to do unnecessary label lookup when it is known in advance that router will have to route packet.

Using traceroute in MPLS networks

RFC4950 introduces extensions to ICMP protocol for MPLS. The basic idea is that some ICMP messages may carry MPLS "label stack object" (list of labels that were on packet when it caused particular ICMP message). ICMP messages of interest for MPLS are Time Exceeded and Need Fragment.

MPLS label carries not only label value, but also TTL field. When imposing label on IP packet, MPLS TTL is set to value in IP header, when last label is removed from IP packet, IP TTL is set to value in MPLS TTL. Therefore MPLS switching network can be diagnosed by means of traceroute tool that supports MPLS extension.

For example, traceroute from R5 to R1 looks like this:

```
[admin@R5] > /tool traceroute 9.9.9.1 src-address=9.9.9.5
     ADDRESS                                    STATUS
   1         4.4.4.3 15ms 5ms 5ms
                      mpls-label=17
   2         2.2.2.2 5ms 3ms 6ms
                      mpls-label=17
   3         9.9.9.1 6ms 3ms 3ms
```

Traceroute results show MPLS labels on packet when it produced ICMP Time Exceeded. The above means: when R3 received packet with MPLS TTL 1, it had label 17 on. This matches label advertised by R3 for 9.9.9.1/32. The same way R2 observed label 17 on packet on next traceroute iteration - R3 switched label 17 to label 17, as explaned above. R1 received packet without labels - R2 did penultimate hop popping as explaned above.

Drawbacks of using traceroute in MPLS network

Label switching ICMP errors

One of drawbacks of using traceroute in MPLS networks is the way MPLS handles produced ICMP errors. In IP networks ICMP errors are simply routed back to source of packet that caused the error. In MPLS network it is possible that router that produces error message does not even have route to source of IP packet (for example in case of assymetric label switching paths or some kind of MPLS tunneling, e.g. to transport MPLS VPN traffic).

Due to this produced ICMP errors are not routed to the source of packet that caused the error, but switched further along label switching path, assuming that when label switching path endpoint will receive ICMP error, it will know how to properly route it back to source.

This causes the situation, that traceroute in MPLS network can not be used the same way as in IP network - to determine failure point in the network. If label switched path is broken anywhere in the middle, no ICMP replies will come back, because they will not make it to the far endpoint of label switching path.

Penultimate hop popping and traceroute source address

Thorough understanding of pen ultimate hop behaviour and routing is necessary to understand and avoid problems that penultimate hop popping causes to traceroute.

In the example setup, regular traceroute from R5 to R1 would yield the following results:

```
[admin@R5] > /tool traceroute 9.9.9.1
     ADDRESS                                  STATUS
   1         0.0.0.0 timeout timeout timeout
   2         2.2.2.2 37ms 4ms 4ms
                      mpls-label=17
   3         9.9.9.1 4ms 2ms 11ms
```

compared to:

```
[admin@R5] > /tool traceroute 9.9.9.1 src-address=9.9.9.5
     ADDRESS                                  STATUS
   1         4.4.4.3 15ms 5ms 5ms
                      mpls-label=17
   2         2.2.2.2 5ms 3ms 6ms
                      mpls-label=17
   3         9.9.9.1 6ms 3ms 3ms
```

The reason why first traceroute does not get response from R3 is that by default traceroute on R5 uses source address 4.4.4.5 for its probes, because it is preferred source for route over which next hop to 9.9.9.1/32 is reachable:

```
[admin@R5] > /ip route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        G GATEWAY         DISTANCE
INTERFACE
 ...
 3 ADC  4.4.4.0/24         4.4.4.5                           0
ether1
 ...
 5 ADo  9.9.9.1/32                         r 4.4.4.3         110
ether1
 ...
```

When first traceroute probe is transmitted (source: 4.4.4.5, destination 9.9.9.1), R3 drops it and produces ICMP error message (source 4.4.4.3 destination 4.4.4.5) that is switched all the way to R1. R1 then sends ICMP error back - it gets switched along label switching path to 4.4.4.5.

R2 is penultimate hop popping router for network 4.4.4.0/24, because 4.4.4.0/24 is directly connected to R3. Therefore R2 removes last label and sends ICMP error to R3 unlabelled:

```
[admin@R2] > /mpls forwarding-table print
 # IN-LABEL             OUT-LABELS           DESTINATION
INTERFACE            NEXTHOP
 ...
 3 19                                        4.4.4.0/24         ether2
2.2.2.3
 ...
```

R3 drops received IP packet, because it receives packet with its own address as source address. ICMP errors produced by following probes come back correctly, because R3 receives unlabelled packets with source addresses 2.2.2.2 and 9.9.9.1, which are acceptable to route.

Command:

```
[admin@R5] > /tool traceroute 9.9.9.1 src-address=9.9.9.5
 ...
```

produces expected results, because source address of traceroute probes is 9.9.9.5. When ICMP errors are travelling back from R1 to R5, penultimate hop popping for 9.9.9.5/32 network happens at R3, therefore it never gets to route packet with its own address as source address.

Configuring VPLS

Configuring VPLS interfaces

VPLS interface can be considered tunnel interface just like EoIP interface. To achieve transparent ethernet segment forwarding between customer sites the following tunnels need to be established:

- R1-R5 (customer A)
- R1-R4 (customer A)
- R4-R5 (customer A)
- R1-R5 (customer B)

Note that each tunnel setup involves creating VPLS interfaces on both endpoints of tunnel.

Negotiation of VPLS tunnels is done by LDP protocol - both endpoints of tunnel exchange labels they are going to use for tunnel. Data forwarding in tunnel then happens by imposing 2 labels on packets: tunnel label and transport label - label that ensures traffic delivery to the other endpoint of tunnel.

VPLS tunnels are configured in /interface vpls menu. vpls-id parameter identifies VPLS tunnel and must be unique for every tunnel between this and remote peer.

The necessary setup:

- on R1:

```
/interface vpls add name=A1toA2 remote-peer=9.9.9.5 mac-
address=00:00:00:00:00:a1 vpls-id=10 disabled=no
/interface vpls add name=A1toA3 remote-peer=9.9.9.4 mac-
address=00:00:00:00:00:a1 vpls-id=10 disabled=no
/interface vpls add name=B1toB2 remote-peer=9.9.9.5 mac-
address=00:00:00:00:00:b1 vpls-id=11 disabled=no
```

- on R4:

```
/interface vpls add name=A3toA1 remote-peer=9.9.9.1 mac-
address=00:00:00:00:00:a3 vpls-id=10 disabled=no
/interface vpls add name=A3toA2 remote-peer=9.9.9.5 mac-
address=00:00:00:00:00:a3 vpls-id=10 disabled=no
```

- on R5:

```
/interface vpls add name=A2toA1 remote-peer=9.9.9.1 mac-
address=00:00:00:00:00:a2 vpls-id=10 disabled=no
```

```
/interface vpls add name=A2toA3 remote-peer=9.9.9.4 mac-
address=00:00:00:00:00:a2 vpls-id=10 disabled=no
/interface vpls add name=B2toB1 remote-peer=9.9.9.1 mac-
address=00:00:00:00:00:b2 vpls-id=11 disabled=no
```

Configuring VPLS tunnel causes dynamic LDP neighbor to be created and "targeted" LDP session to be established. Targeted LDP session is session that is established between two routers that are not direct neighbors. After this setup R1 LDP neighbors are:

```
[admin@R1] > mpls ldp neighbor pr
Flags: X - disabled, D - dynamic, O - operational, T - sending-
targeted-hello, V - vpls
 #      TRANSPORT       LOCAL-TRANSPORT PEER
SEND-TARGETED ADDRESSES
 0 DO   9.9.9.2         9.9.9.1         9.9.9.2:0                    no
1.1.1.2

2.2.2.2

9.9.9.2
 1 DOTV 9.9.9.5         9.9.9.1         9.9.9.5:0
yes           4.4.4.5

5.5.5.5

9.9.9.5
 2 DOTV 9.9.9.4         9.9.9.1         9.9.9.4:0
yes           3.3.3.4

5.5.5.4

9.9.9.4
```

Note that labels for IP routes are also exchanged between VPLS peers, although there is no chance any of them will be used. For example, without adding additional links, R4 will not become next hop for any route on R1, so labels learned from R4 are not likely to be ever used. Still routers maintain all labels exchanged so that they are ready for use immediately if needed. This default behaviour can be overridden by filtering which is discussed later.

By monitoring state of VPLS interface its related information can be viewed:

```
[admin@R1] /interface vpls> monitor A1toA3 once
    remote-label: 24
     local-label: 27
   remote-status:
      igp-prefix: 9.9.9.4/32
     igp-nexthop: 1.1.1.2
  imposed-labels: 21,24
```

Here we can see that R1 has assigned label 27 for tunnel between R1 and R4. MPLS forwarding table shows that this label is recognized and instead of fowarding to some next hop, received over this tunnel:

```
[admin@R1] > /mpls forwarding-table print
 # IN-LABEL             OUT-LABELS           DESTINATION
INTERFACE          NEXTHOP
 ...
11 27                                        A1toA3
 ...
```

In turn remote endpoint (R4) has assigned label 24.

igp-prefix shows route that is used to get to remote endpoint of tunnel. This implies that when forwarding traffic to remote endpoint of tunnel this router will impose transport label - label distributed by next hop (which is shown as igp-nexthop) to 9.9.9.4/32 for 9.9.9.4/32 route. This can be confirmed on R2:

```
[admin@R2] > /mpls forwarding-table print
 # IN-LABEL             OUT-LABELS           DESTINATION
INTERFACE          NEXTHOP
 ...
 5 21                   18                   9.9.9.4/32         ether2
2.2.2.3
 ...
```

Tunnel label imposed on packets will be as assigned by remote router (R4) for this tunnel.

imposed-labels reflect this setup: packets produced by tunnel will have 2 labels on them: 21 and 24.

Penultimate hop popping effects on VPLS tunnels

Penultimate hop popping of transport label causes packets to arrive at VPLS tunnel endpoint with just one tag - tunnel tag. This makes VPLS tunnel endpoint do just one label lookup to find out what to do with packet. Transport label behaviour can be observed by traceroute tool between tunnel endpoints. For example, traceroute from R1 to R4 looks like this:

```
[admin@R1] > /tool traceroute 9.9.9.4 src-address=9.9.9.1
     ADDRESS                                    STATUS
   1         1.1.1.2 7ms 5ms 3ms
                      mpls-label=21
   2         2.2.2.3 5ms 4ms 18ms
                      mpls-label=18
```

```
   3          9.9.9.4 4ms 5ms 3ms
```

traceroute output shows, that endpoint of tunnel is receiving probe without label. The same happens with VPLS tunnel traffic - at R3 transport label (18) is popped and packet is switched with just tunnel label on.

The requirement to deliver packet with tunnel label to endpoint of tunnel explains configuration advice to use "loopback" IP addresses as tunnel endpoints. If in this case R4 was establishing LDP sessions with its address 3.3.3.4, penultimate hop popping would happen not at R3, but at R2, because R3 has network 3.3.3.0/24 as its connected network (and therefore advertises implicit null label). This would cause R3 (and not R4) to receive packet with just tunnel label on, yielding unpredicted results - either dropping frame if R3 does not recognize the packet or forwarding it the wrong way.

Another issue is having VPLS tunnel endpoints directly connected, as in case of R4 and R5. There are no transport labels they can use between themselves, because they both instruct other one to be penultimate hop popping router for their tunnel endpoint address. For example on R5:

```
[admin@R5] > /mpls remote-bindings print
Flags: X - disabled, A - active, D - dynamic
 #    DST-ADDRESS        NEXTHOP         LABEL      PEER
 ...
 3 AD 9.9.9.4/32         5.5.5.4         impl-null  9.9.9.4:0
 ...
```

This causes VPLS tunnel to use only tunnel label when sending packets:

```
[admin@R5] > /int vpls monitor A2toA3 once
    remote-label: 23
     local-label: 27
   remote-status:
      igp-prefix: 9.9.9.4/32
     igp-nexthop: 5.5.5.4
  imposed-labels: 23
```

Bridging ethernet segments with VPLS

VPLS tunnels provide virtual ethernet link between routers. To transparrently connect two physical ethernet segments, they must be bridged with VPLS tunnel. In general it gets done the same way as with EoIP interfaces.

So to transparently bridge customer B networks connected to R1 and R5 the following commands are used on R1:

```
/interface bridge add name=B
/interface bridge port add bridge=B interface=ether1
```

```
/interface bridge port add bridge=B interface=B1toB2
```

and on R5:

```
/interface bridge add name=B
/interface bridge port add bridge=B interface=ether3
/interface bridge port add bridge=B interface=B2toB1
```

Note that there is not need to run (R)STP protocol on bridge as there are links between segments B1 and B2 except single VPLS tunnel between R1 and R5.

Split horizon bridging

In the example setup there are 3 tunnels set up to connect segments A1, A2 and A3, establishing so called "full mesh" of tunnels between involved segments. If bridging without (R)STP was enabled, traffic loop would occur. There are a few solutions to this:

- enabling (R)STP to eliminate the loop. This approach has a drawback - (R)STP protocol would disable forwarding through one of tunnels and keep it just for backup purposes. That way traffic between 2 segments would have to go through 2 tunnels, making setup inefficent
- using bridge firewall to make sure that traffic does not get looped - involves firewall rule setup making bridging less efficent
- using bridge horizon feature

The basic idea of split horizon bridging is to make traffic arriving over some port never be sent out some set of ports. For VPLS purposes this would mean never sending packet that arrived over one VPLS tunnel over to other VPLS tunnel, as it is known in advance, that sender of packet has connection to target network itself.

For example, if device in A1 sent packet to broadcast or unknown MAC address (which causes bridges to flood all interfaces), it would get sent to both, R5 and R4 over VPLS tunnels. In regular setup e.g. R5 when receiving such packet over VPLS tunnel would send it in A2 connected to it and also over VPLS tunnel to R4. This way R4 would get 2 copies of the same packet and further cause traffic to loop.

Bridge horizon feature allows to configure bridge ports with horizon setting so that packet received over port with horizon value X is not forwarded or flooded to any port with the same horizon value X. So in case of full mesh of VPLS tunnels, each router must be configured with the same horizon value for VPLS tunnels that are bridged together.

For example, configuration commands for R1 to enable bridging for customer A are:

```
/interface bridge add name=A
/interface bridge port add bridge=A interface=A1toA2 horizon=1
```

```
/interface bridge port add bridge=A interface=A1toA3 horizon=1
```

In similar way bridge should be configured on R4 and R5. Note that physical ethernet port is not configured with horizon value. If it was, that would disabled bridge forwarding data at all.

Note that horizon value has meaning only locally - it does not get transmitted over network, therefore it does not matter if the same value is configured in all routers participating in bridged network.

Optimizing label distribution

Label binding filtering

During implementation of given example setup, it has become clear that not all label bindings are necessary. For example, there is no need to exchange IP route label bindings between R1 and R5 or R1 and R4, as there is no chance they will ever be used. Also, if given network core is providing connectivity only for mentioned customer ethernet segments, there is no real use to distribute labels for networks that connect routers between themselves, the only routes that matter are /32 routes to endpoints of VPLS tunnels.

Label binding filtering can be used to distribute only specified sets of labels to reduce resource usage and network load.

There are 2 types of label binding filters:

- which label bindings should be advertised to LDP neighbors, configured in /mpls ldp advertise-filter
- which label bindings should be accepted from LDP neighbors, configured in /mpls ldp accept-filter

Filters are organized in ordered list, specifying prefix that must include the prefix that is tested against filter and neighbor (or wildcard).

In given example setup all routers can be configured so that they advertise labels only for routes that allow to reach endpoints of tunnels. For this 2 advertise filters need to be configured on all routers:

```
/mpls ldp advertise-filter add prefix=9.9.9.0/24 advertise=yes
/mpls ldp advertise-filter add prefix=0.0.0.0/0 advertise=no
```

This filter causes routers to advertise only bindings for routes that are included by 9.9.9.0/24 prefix which covers tunnel endpoints (9.9.9.1/32, 9.9.9.4/32, 9.9.9.5/32). The second rule is necessary because default filter result, when no rule matches is to allow action in question.

In given setup there is no need to set up accept filter, because by convention introduced by 2 abovementioned rules no LDP router will distribute unnecessary bindings.

Note that filter changes do not affect existing mappings, so to take filter into effect, connections between neighbors need to be reset. This can get done by removing them:

```
[admin@R1] /mpls ldp neighbor> print
Flags: X - disabled, D - dynamic, O - operational, T - sending-
targeted-hello, V - vpls
 #      TRANSPORT       LOCAL-TRANSPORT PEER
SEND-TARGETED ADDRESSES
 0 DO   9.9.9.2         9.9.9.1         9.9.9.2:0                    no
1.1.1.2

2.2.2.2

9.9.9.2
 1 DOTV 9.9.9.5         9.9.9.1         9.9.9.5:0
yes           4.4.4.5

5.5.5.5

9.9.9.5
 2 DOTV 9.9.9.4         9.9.9.1         9.9.9.4:0
yes           3.3.3.4

5.5.5.4

9.9.9.4
[admin@R1] /mpls ldp neighbor> remove [find]
```

So on R1, for example we get:

```
[admin@R1] > /mpls remote-bindings print
Flags: X - disabled, A - active, D - dynamic
 #    DST-ADDRESS        NEXTHOP         LABEL      PEER
 0  D 9.9.9.1/32                         30         9.9.9.5:0
 1  D 9.9.9.5/32                         impl-null  9.9.9.5:0
 2  D 9.9.9.4/32                         31         9.9.9.5:0
 3  D 9.9.9.2/32                         32         9.9.9.5:0
 4  D 9.9.9.3/32                         33         9.9.9.5:0
 5 AD 9.9.9.2/32         1.1.1.2         impl-null  9.9.9.2:0
 6  D 9.9.9.1/32                         24         9.9.9.2:0
 7 AD 9.9.9.3/32         1.1.1.2         25         9.9.9.2:0
 8 AD 9.9.9.4/32         1.1.1.2         26         9.9.9.2:0
 9 AD 9.9.9.5/32         1.1.1.2         27         9.9.9.2:0
```

```
10  D 9.9.9.1/32                          27         9.9.9.4:0
11  D 9.9.9.5/32                          28         9.9.9.4:0
12  D 9.9.9.4/32                          impl-null  9.9.9.4:0
13  D 9.9.9.2/32                          29         9.9.9.4:0
14  D 9.9.9.3/32                          30         9.9.9.4:0
```

There still are unnecessary bindings, this time - the bindings distributed due to establishing targeted LDP session with remote endpoints of VPLS tunnels (bindings from 9.9.9.5 and 9.9.9.4)

To filter out those, we configure routers to not distribute any IP bindings to any of tunnel endpoint routers. For example on R1, filter table should look like this:

```
[admin@R1] /mpls ldp advertise-filter> print
Flags: X - disabled
 #   PREFIX             NEIGHBOR        ADVERTISE
 0   0.0.0.0/0          9.9.9.4         no
 1   0.0.0.0/0          9.9.9.5         no
 2   9.9.9.0/24         all             yes
 3   0.0.0.0/0          all             no
```

This causes routers to have minimal label binding tables, for example on R1:

```
[admin@R1] > /mpls local-bindings print
Flags: X - disabled, A - advertised, D - dynamic, L - local-route, G -
gateway-route, e - egress
 #      DST-ADDRESS        LABEL      PEERS
 0 ADLe 9.9.9.1/32         impl-null  9.9.9.2:0
 1 ADG  9.9.9.3/32         40         9.9.9.2:0
 2 ADG  9.9.9.4/32         41         9.9.9.2:0
 3 ADG  9.9.9.2/32         42         9.9.9.2:0
 4 ADG  9.9.9.5/32         43         9.9.9.2:0
[admin@R1] > /mpls remote-bindings print
Flags: X - disabled, A - active, D - dynamic
 #    DST-ADDRESS        NEXTHOP         LABEL      PEER
 0 AD 9.9.9.2/32         1.1.1.2         impl-null  9.9.9.2:0
 1  D 9.9.9.1/32                         24         9.9.9.2:0
 2 AD 9.9.9.3/32         1.1.1.2         25         9.9.9.2:0
 3 AD 9.9.9.4/32         1.1.1.2         26         9.9.9.2:0
 4 AD 9.9.9.5/32         1.1.1.2         27         9.9.9.2:0
```

Note that IP binding distribution should not be disabled between R4 and R5 although they are tunnel endpoints. Doing so would not harm regular case, because R4 and R5 does not need IP bindings to VPLS tunnel data, but in case link between R3 and R5 would go down, all traffic to R5 from R1 would have to be rerouted through R4. In such case R5 not distributing IP bindings to R4 would cause R4 to not be able to forward MPLS traffic to R5.

Effects of label binding filtering on data forwarding in network

Note the traceroute results after these changes. Traceroute from R1 to R5 using R1 loopback address as source address still behaves the same - each hop reports received labels:

```
[admin@R1] > tool traceroute 9.9.9.5 src-address=9.9.9.1
     ADDRESS                                  STATUS
   1         1.1.1.2 11ms 4ms 5ms
                      mpls-label=27
   2         2.2.2.3 4ms 4ms 4ms
                      mpls-label=25
   3         9.9.9.5 12ms 3ms 3ms
```

But in case of traceroute using R1 address that faces network:

```
[admin@R1] > tool traceroute 9.9.9.5 src-address=1.1.1.1
     ADDRESS                                  STATUS
   1         0.0.0.0 timeout timeout timeout
   2         0.0.0.0 timeout timeout timeout
   3         9.9.9.5 3ms 3ms 3ms
```

Now all hops except the last one do not respond. The reason for this is the fact, that there is no label switching path back from R5 to R1 (which this time uses address 1.1.1.1) because there are no label bindings distributed - so ICMP response is routed and routers on the way back (R3 and R2) receive packets with their own source address and drop them right away without routing.

On the other hand, traceroute from R1 to R5 using their non-loopback addresses:

```
[admin@R1] > tool traceroute 4.4.4.5 src-address=1.1.1.1
     ADDRESS                                  STATUS
   1         1.1.1.2 13ms 1ms 1ms
   2         2.2.2.3 3ms 2ms 2ms
   3         4.4.4.5 3ms 3ms 23ms
```

There is no label switching involved doing this traceroute and it works just like in network without MPLS at all.

HauteRouterOS Multicast Routing Protocol

Document revision: 2.2 (Tuesday, October 30, 2007)

Applies to: HauteRouterOS V3.0

HauteSpot is using PIM-SM protocol implementation from XORP, integrated in our own routing program.

Multicast Routing Overview

IP Multicast is a technology that allows one-to-many and many-to-many distribution of data on the Internet. Senders send their data to a multicast IP destination address, and receives express an interest in receiving traffic destined for such an address. The network then figures out how to get the data from senders to receivers.

If both the sender and receiver for a multicast group are on the same local broadcast subnet, then the routers do not need to be involved in the process, and communication can take place directly. If, however, the sender and receiver are on different subnets, then a multicast routing protocol needs to be involved in setting up multicast forwarding state on the tree between the sender and the receivers.

HauteSpot supports PIM-SM multicast routing protocol. PIM means "platform independent multicast" - i.e. this protocol is not tied to any particular unicast routing IGP. SM means "sparse-mode"; as opposed to dense-mode, in sparse-mode protocols explicit control messages are used to ensure that traffic is only delivered to the subnets where there are receivers that requested to receive it.

In addition to the routing protocols used to set up forwarding state between subnets, a way is needed for the routers to discover that there are local receivers on a directly attached subnet. For IPv4 this role is served by the Internet Group Management Protocol (IGMP).

Service Models: ASM vs SSM

There are two different models for IP multicast:

- Any Source Multicast (ASM), in which a receiver joins a multicast group, and receives traffic from any senders that send to that group.
- Source-Specific Multicast (SSM), in which a receiver explicitly joins to a (source, group) pairing.

Multicast Addressing

For IPv4, multicast addresses are in the range 224.0.0.0 to 239.255.255.255 inclusive. Addresses within 224.0.0.0/24 are considered link-local and should not be forwarded between subnets. Addresses within 232.0.0.0/8 are reserved for SSM usage. Addresses in 239.0.0.0/8 are ASM addresses defined for varying sizes of limited scope.

IGMP

When a receiver joins a multicast group, the multicast routers serving that receiver's subnet need to know that the receiver has joined so that they can arrange for multicast traffic destined for that group to reach this subnet. The Internet Group Management Protocol (IGMP) is a link-local protocol for IPv4 that communicates this information between receivers and routers. The same role for IPv6 is performed by the Multicast Listener Discovery protocol (MLD).

The basic IGMP mechanism works as follows. When a multicast receiver joins a multicast group it multicasts an IGMP Join message onto the subnet on which it is

joining. The local routers receive this join, and cause multicast traffic destined for the group to reach this subnet. Periodically one of the local routers sends a IGMP Query message onto the subnet. If there are multiple multicast routers on the subnet, then one of them is elected as the sole querier for that subnet. In response to an IGMP query, receivers respond by refreshing their IGMP Join. If the join is not refreshed in response to queries, then the state is removed, and multicast traffic for this group ceases to reach this subnet.

There are three different versions of IGMP:

- IGMP version 1 functions as described above.
- IGMP version 2 adds support for IGMP Leave messages to allow fast leave from a multicast group.
- IGMP version 3 adds support for source include and exclude lists, to allow a receiver in indicate that it only wants to hear traffic from certain sources, or not receive traffic from certain sources.

PIM-SM Protocol Overview

From the PIM-SM specification:

PIM-SM relies on an underlying topology-gathering protocol to populate a routing table with routes. This routing table is called the MRIB or Multicast Routing Information Base. The routes in this table may be taken directly from the unicast routing table, or it may be different and provided by a separate routing protocol such as Multi-protocol BGP.

Regardless of how it is created, the primary role of the MRIB in the PIM-SM protocol is to provide the next-hop router along a multicast-capable path to each destination subnet. The MRIB is used to determine the next-hop neighbor to which any PIM Join/Prune message is sent. Data flows along the reverse path of the Join messages. Thus, in contrast to the unicast RIB which specifies the next-hop that a data packet would take to get to some subnet, the MRIB gives reverse-path information, and indicates the path that a multicast data packet would take from its origin subnet to the router that has the MRIB.

Phase One: RP Tree

In phase one, a multicast receiver expresses its interest in receiving traffic destined for a multicast group. Typically it does this using IGMP or MLD. One of the receiver's local PIM routers is elected as the Designated Router (DR) for that subnet. On receiving the receiver's expression of interest, the DR then sends a PIM Join message towards the Rendezvous Point (RP) for that multicast group. The RP is a PIM-SM router that has been configured to serve a bootstrapping role for certain multicast groups. This Join message is known as a (*,G) Join because it joins group G for all sources to that group. The (*,G) Join travels hop-by-hop towards the RP for the group, and in each router it passes through, multicast tree state for group G is instantiated. Eventually the (*,G) Join either reaches the RP, or reaches a router that already has (*,G) Join state for that group. When many receivers join the group, their Join messages converge on the RP, and form a distribution tree for group G that is rooted at the RP. This is known as the RP Tree (RPT), and is also known as

the shared tree because it is shared by all sources sending to that group. Join messages are resent periodically so long a the receiver remains in the group. When all receivers on a leaf-network leave the group, the DR will send a PIM (*,G) Prune message towards the RP for that multicast group. However if the Prune message is not sent for any reason, the state will eventually time out.

A multicast data sender just starts sending data destined for a multicast group. The sender's local router (DR) takes those data packets, unicast-encapsulates them, and sends them directly to the RP. The RP receives these encapsulated data packets, decapsulates them, and forwards them onto the shared tree. The packets then follow the (*,G) multicast tree state in the routers on the RP Tree, being replicated wherever the RP Tree branches, and eventually reaching all the receivers for that multicast group. The process of encapsulating data packets to the RP is called registering, and the encapsulation packets are known as PIM Register packets. At the end of phase one, multicast traffic is flowing encapsulated to the RP, and then natively over the RP tree to the multicast receivers.

Phase Two: Register-Stop

Register-encapsulation of data packets is inefficient for two reasons:

- Encapsulation and decapsulation may be relatively expensive operations for a router to perform, depending on whether or not the router has appropriate hardware for these tasks.
- Traveling all the way to the RP, and then back down the shared tree may entail the packets traveling a relatively long distance to reach

receivers that are close to the sender. For some applications, this increased latency is undesirable.

Although Register-encapsulation may continue indefinitely, for the reasons above, the RP will normally choose to switch to native forwarding. To do this, when the RP receives a register-encapsulated data packet from source S on group G, it will normally initiate an (S,G) source-specific Join towards S. This Join message travels hop-by-hop towards S, instantiating (S,G) multicast tree state in the routers along the path. (S,G) multicast tree state is used only to forward packets for group G if those packets come from source S. Eventually the Join message reaches S's subnet or a router that already has (S,G) multicast tree state, and then packets from S start to flow following the (S,G) tree state towards the RP. These data packets may also reach routers with (*,G) state along the path towards the RP - if so, they can short-cut onto the RP tree at this point.

While the RP is in the process of joining the source-specific tree for S, the data packets will continue being encapsulated to the RP. When packets from S also start to arrive natively at the the RP, the RP will be receiving two copies of each of these packets. At this point, the RP starts to discard the encapsulated copy of these packets, and it sends a Register-Stop message back to S's DR to prevent the DR unnecessarily encapsulating the packets.

At the end of phase 2, traffic will be flowing natively from S along a source-specific tree to the RP, and from there along the shared tree to the receivers. Where the two

trees intersect, traffic may transfer from the source-specific tree to the RP tree, and so avoid taking a long detour via the RP. It should be noted that a sender may start sending before or after a receiver joins the group, and thus phase two may happen before the shared tree to the receiver is built.

Phase 3: Shortest-Path Tree

Although having the RP join back towards the source removes the encapsulation overhead, it does not completely optimize the forwarding paths. For many receivers the route via the RP may involve a significant detour when compared with the shortest path from the source to the receiver. To obtain lower latencies, a router on the receiver's LAN, typically the DR, may optionally initiate a transfer from the shared tree to a source-specific shortest-path tree (SPT). To do this, it issues an (S,G) Join towards S. This instantiates state in the routers along the path to S. Eventually this join either reaches S's subnet, or reaches a router that already has (S,G) state. When this happens, data packets from S start to flow following the (S,G) state until they reach the receiver.

At this point the receiver (or a router upstream of the receiver) will be receiving two copies of the data - one from the SPT and one from the RPT. When the first traffic starts to arrive from the SPT, the DR or upstream router starts to drop the packets for G from S that arrive via the RP tree. In addition, it sends an (S,G) Prune message towards the RP. This is known as an (S,G,rpt) Prune. The Prune message travels hop-by-hop, instantiating state along the path towards the RP indicating that traffic from S for G should NOT be forwarded in this direction. The prune is propagated until it reaches the RP or a router that still needs the traffic from S for other receivers. By now, the receiver will be receiving traffic from S along the shortest-path tree between the receiver and S. In addition, the RP is receiving the traffic from S, but this traffic is no longer reaching the receiver along the RP tree. As far as the receiver is concerned, this is the final distribution tree.

Multi-access Transit LANs

Additional problems here (as oppossed to PtP interfaces) are solved by introduction of PIM Assert messages... (*TODO*)

RP Discovery

PIM-SM routers need to know the address of the RP for each group for which they have (*,G) state. This address is obtained either through a bootstrap mechanism or through static configuration. One dynamic way to do this is to use the Bootstrap Router (BSR) mechanism. One router in each PIM-SM domain is elected the Bootstrap Router through a simple election process. All the routers in the domain that are configured to be candidates to be RPs periodically unicast their candidacy to the BSR. From the candidates, the BSR picks an RP-set, and periodically announces this set in a Bootstrap message. Bootstrap messages are flooded hop-by-hop throughout the domain until all routers in the domain know the RP-Set. To map a group to an RP, a router hashes the group address into the RP-set using an order-preserving hash function (one that minimizes changes if the RP-Set changes). The resulting RP is the one that it uses as the RP for that group.

Configuration Syntax

General Settings

switch-to-spt - this directive permits the specification of a bitrate threshold at a lasthop router or RP for switching from the RP Tree to the Shortest-Path Tree.

switch-to-spt-interval - this specifies the measurement interval in seconds for measuring the bitrate of traffic from a multicast sender. The measurement interval should normally not be set too small. The default value is 100 seconds.

switch-to-spt-bytes - this specifies the maximum number of bytes from a multicast sender that can be received in interval seconds. If this threshold is exceeded, the router will attempt to switch to the shortest-path tree from that multicast sender.

PIM Interfaces

Allows to enable and configure PIM and IGMP protocols on a specific interface. PIM will be active only if interface configuration is present.

Multicast Routing Information Base (MRIB)

MRIB routes are used for reverse path forwarding check. In a way, they perform opposite function that FIB (*Forwarding Information Base*) routes: FIB is used to find the right By default, MRIB is populated by FIB routes. Use "multicast" routing filter chanin to control that or set specific paramteres for imported FIB routes (e.g. you can change the distance of the route). In addition, you can specify static MRIB routes. This is useful only if you are usign multihoming and multicast packet flow will be different from unicast packet flow.

Active MRIB entries that are imported from FIB are shown with "dynamic" flag.

Rendezvous Points (RP)

Static RP configuration.

Bootstrap (bsr-candidates, rp-candidates, bsr)

Bootstrap protocol configuration.

Neighbors

Read-only status information about neighbor routers running PIM.

Join

Read-only status information about active IGMP joins.

MFC

Read-only status information about the Multicast Forwarding Cache.

Example

Almost minimal setup where multicast routing is neccessary:

- multicast sender (server);
- multicast receiver (client);
- two routers running PIM beteen them.

Multicast traffic in this example will be destined to address 224.0.1.20
Traffic flow:

```
 Sender -- (subnet I) --> Router A -- (subnet II) --> Router B -- (subnet III) --> Receiver
```

Router A will be configured as Rendezvous Point.
Enable PIM and IGMP router A:

```
[admin@A] > routing pim interface add
[admin@A] > routing pim interface p
Flags: X - disabled, I - inactive, D - dynamic, R - designated-router,
v1 - IGMPv1, v2 - IGMPv2, v3 - IGMPv3
 #     INTERFACE    PROTOCOLS
 0   v2 all          pim
                     igmp
 1 DRv2 ether3       pim
                     igmp
 2 DR   register     pim
```

Configure static Rendezvous Point:

```
[admin@A] > routing pim rp add address=<IP of router A>
```

Enable PIM and IGMP router B:

```
[admin@B] > routing pim interface add interface=ether1
[admin@B] > routing pim interface p
Flags: X - disabled, I - inactive, D - dynamic, R - designated-router,
v1 - IGMPv1, v2 - IGMPv2, v3 - IGMPv3
 #     INTERFACE    PROTOCOLS
 0  Rv2 ether1       pim
                     igmp
 1 DR   register     pim
```

Configure static Rendezvous Point:

```
[admin@B] > routing pim rp add address=<IP of router A>
```

Add route on multicast sender:

```
# ip route add 224.0.1.20/32 via <IP of router A>
```

Start sender and receiver programs. You can either write simple programs yourself, or use any of these:

- MGEN is a program that can be used to send or receive multicast traffic, etc: http://cs.itd.nrl.navy.mil/work/mgen/index.php
- Alternatively, mtest is a bare-bones sender/receiver program available from: http://netweb.usc.edu/pim/pimd/

And hey, it works! Client should receive data now.

Caveats

- Route metric cannot be configured, 0xffff is always used instead (important for PIM asserts). Route distance (from FIB or static MRIB) is used as "metric preference" and can be only in range 0..255.
- Scope zones are not supported.
- It is unclear whether Linux kernel fully supports IGMPv3.

FAQ

Q. Does HauteSpot support Source Specific Multicast (SSM)?
A. Yes, SSM is a part of PIM-SM specification and we support it.
Q. Is support for PIM-DM planned?
A. No, as PIM-SM performs good in almost every setup, both sparse and dense.

References

1. XORP User Manual, chapters 11 - 14
2. Multicast tutorial. Deals with multicast addressing IGMP, PIM-SM / SSM, MSDP and MBGP
3. RFC 2236: Internet Group Management Protocol, Version 2
4. RFC 3376: Internet Group Management Protocol, Version 3
5. RFC 4601: Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)
6. Internet draft: Bootstrap Router (BSR) Mechanism for PIM

d-only: time) - seconds elapsed since last received packet.

HauteRouterOS NAT

Document revision: 2.7 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

Network Address Translation (NAT) is a router facility that replaces source and (or) destination IP addresses of the IP packet as it pass through the router. It is most commonly used to enable multiple host on a private network to access the Internet using a single public IP address.

Specifications

Packages required: ***system***
License required: *Level1 (number of rules limited to 1)* , *Level3*
Submenu level: ***/ip firewall nat***
Standards and Technologies: IP, RFC1631, RFC2663
Hardware usage: *Increases with the count of rules*

NAT

Description

Network Address Translation is an Internet standard that allows hosts on local area networks to use one set of IP addresses for internal communications and another set of IP addresses for external communications. A LAN that uses NAT is referred as *natted* network. For NAT to function, there should be a NAT gateway in each natted network. The NAT gateway (NAT router) performs IP address rewriting on the way a packet travel from/to LAN.

There are two types of NAT:

- source NAT or srcnat. This type of NAT is performed on packets that are originated from a natted network. A NAT router replaces the private source address of an IP packet with a new public IP address as it travels through the router. A reverse operation is applied to the reply packets travelling in the other direction.
- destination NAT or dstnat. This type of NAT is performed on packets that are destined to the natted network. It is most comonly used to make hosts on a private network to be acceesible from the Internet. A NAT router performing dstnat replaces the destination IP address of an IP packet as it travel through the router towards a private network.

NAT Drawbacks

Hosts behind a NAT-enabled router do not have true end-to-end connectivity. Therefore some Internet protocols might not work in scenarios with NAT. Services that require the initiation of TCP connection from outside the private network or stateless protocols such as UDP, can be disrupted. Moreover, some protocols are inherently incompatible with NAT, a bold example is AH protocol from the IPsec suite.

RouterOS includes a number of so-called NAT helpers, that enable NAT traversal for various protocols.

Redirect and Masquerade

Redirect and masquerade are special forms of destination NAT and source NAT, respectively. Redirect is similar to the regular destination NAT in the same way as masquerade is similar to the source NAT - masquerade is a special form of source NAT without need to specify **to-addresses** - outgoing interface address is used automatically. The same is for redirect - it is a form of destination NAT where **to-addresses** is not used - incoming interface address is used instead. Note that **to-ports** is meaningful for redirect rules - this is the port of the service on the router that will handle these requests (e.g. web proxy).

When packet is dst-natted (no matter - **action=nat** or **action=redirect**), dst address is changed. Information about translation of addresses (including original dst address) is kept in router's internal tables. Transparent web proxy working on router (when web requests get redirected to proxy port on router) can access this information from internal tables and get address of web server from them. If you are dst-natting to some different proxy server, it has no way to find web server's address from IP header (because dst address of IP packet that previously was address of web server has changed to address of proxy server). Starting from HTTP/1.1 there is special header in HTTP request which tells web server address, so proxy server can use it, instead of dst address of IP packet. If there is no such header (older HTTP version on client), proxy server can not determine web server address and therefore can not work.

It means, that it is impossible to correctly transparently redirect HTTP traffic from router to some other transparent-proxy box. Only correct way is to add transparent proxy on the router itself, and configure it so that your "real" proxy is parent-proxy. In this situation your "real" proxy does not have to be transparent any more, as proxy on router will be transparent and will forward proxy-style requests (according to standard; these requests include all necessary information about web server) to "real" proxy.

Property Description

action (accept | add-dst-to-address-list | add-src-to-address-list | dst-nat | jump | log | masquerade | netmap | passthrough | redirect | return | same | src-nat; default: **accept**) - action to undertake if the packet matches the rule
accept - accepts the packet. No action is taken, i.e. the packet is passed through and no more rules are applied to it
add-dst-to-address-list - adds destination address of an IP packet to the address list specified by **address-list** parameter
add-src-to-address-list - adds source address of an IP packet to the address list specified by **address-list** parameter
dst-nat - replaces destination address of an IP packet to values specified by **to-addresses** and **to-ports** parameters
jump - jump to the chain specified by the value of the **jump-target** parameter
log - each match with this action will add a message to the system log
masquerade - replaces source address of an IP packet to an automatically determined by the routing facility IP address
netmap - creates a static 1:1 mapping of one set of IP addresses to another one. Often used to distribute public IP addresses to hosts on private networks
passthrough - ignores this rule goes on to the next one

redirect - replaces destination address of an IP packet to one of the router's local addresses
return - passes control back to the chain from where the jump took place
same - gives a particular client the same source/destination IP address from supplied range for each connection. This is most frequently used for services that expect the same client address for multiple connections from the same client
src-nat - replaces source address of an IP packet to values specified by **to-addresses** and **to-ports** parameters
address-list (*name*) - specifies the name of the address list to collect IP addresses from rules having **action=add-dst-to-address-list** or **action=add-src-to-address-list** actions. These address lists could be later used for packet matching
address-list-timeout (*time*; default: **00:00:00**) - time interval after which the address will be removed from the address list specified by **address-list** parameter. Used in conjunction with **add-dst-to-address-list** or **add-src-to-address-list** actions
00:00:00 - leave the address in the address list forever
chain (dstnat | srcnat | *name*) - specifies the chain to put a particular rule into. As the different traffic is passed through different chains, always be careful in choosing the right chain for a new rule. If the input does not match the name of an already defined chain, a new chain will be created
dstnat - a rule placed in this chain is applied after routing. The rules that replace destination addresses of IP packets should be placed there
srcnat - a rule placed in this chain is applied before routing. The rules that replace the source addresses of IP packets should be placed there
comment (*text*) - a descriptive comment for the rule. A comment can be used to identify rules form scripts
connection-bytes (*integer-integer*) - matches packets only if a given amount of bytes has been transfered through the particular connection
0 - means infinity, *exempli gratia*: **connection-bytes=2000000-0** means that the rule matches if more than 2MB has been transfered through the relevant connection
connection-limit (*integer*,*netmask*) - restrict connection limit per address or address block
connection-mark (*name*) - matches packets marked via mangle facility with particular connection mark
connection-type (ftp | gre | h323 | irc | mms | pptp | quake3 | tftp) - matches packets from related connections based on information from their connection tracking helpers. A relevant connection helper must be enabled under **/ip firewall service-port**
content (*text*) - the text packets should contain in order to match the rule
dst-address (*IP address*/*netmask* | *IP address-IP address*) - specifies the address range an IP packet is destined to. Note that console converts entered **address/netmask** value to a valid network address, i.e.:**1.1.1.1/24** is converted to **1.1.1.0/24**
dst-address-list (*name*) - matches destination address of a packet against user-defined address list
dst-address-type (unicast | local | broadcast | multicast) - matches destination address type of the IP packet, one of the:
unicast - IP addresses used for one point to another point transmission. There is only one sender and one receiver in this case
local - matches addresses assigned to router's interfaces
broadcast - the IP packet is sent from one point to all other points in the IP subnetwork
multicast - this type of IP addressing is responsible for transmission from one or more points to a set of other points
dst-limit (*integer*/*time*{0,1},*integer*,dst-address | dst-port | src-address{+},*time*{0,1}) - limits the packet per second (pps) rate on a per destination

IP or per destination port base. As opposed to the **limit** match, every destination IP address / destination port has it's own limit. The options are as follows (in order of appearance):
Count - maximum average packet rate, measured in packets per second (pps), unless followed by **Time** option
Time - specifies the time interval over which the packet rate is measured
Burst - number of packets to match in a burst
Mode - the classifier(-s) for packet rate limiting
Expire - specifies interval after which recorded IP addresses / ports will be deleted
dst-port (*integer*: 0..65535-*integer*: 0..65535{*}) - destination port number or range
hotspot (*multiple choice:* from-client | auth | local-dst) - matches packets received from clients against various Hot-Spot. All values can be negated
from-client - true, if a packet comes from HotSpot client
auth - true, if a packet comes from authenticted client
local-dst - true, if a packet has local destination IP address
icmp-options (*integer*:*integer*) - matches ICMP Type:Code fields
in-interface (*name*) - interface the packet has entered the router through
ipv4-options (any | loose-source-routing | no-record-route | no-router-alert | no-source-routing | no-timestamp | none | record-route | router-alert | strict-source-routing | timestamp) - match ipv4 header options
any - match packet with at least one of the ipv4 options
loose-source-routing - match packets with loose source routing option. This option is used to route the internet datagram based on information supplied by the source
no-record-route - match packets with no record route option. This option is used to route the internet datagram based on information supplied by the source
no-router-alert - match packets with no router alter option
no-source-routing - match packets with no source routing option
no-timestamp - match packets with no timestamp option
record-route - match packets with record route option
router-alert - match packets with router alter option
strict-source-routing - match packets with strict source routing option
timestamp - match packets with timestamp
jump-target (dstnat | srcnat*name*) - name of the target chain to jump to, if the **action=jump** is used
limit (*integer*/*time*{0,1},*integer*) - restricts packet match rate to a given limit. Usefull to reduce the amount of log messages
Count - maximum average packet rate, measured in packets per second (pps), unless followed by **Time** option
Time - specifies the time interval over which the packet rate is measured
Burst - number of packets to match in a burst
log-prefix (*text*) - all messages written to logs will contain the prefix specified herein. Used in conjunction with **action=log**
nth (*integer*,*integer*: 0..15,*integer*{0,1}) - match a particular Nth packet received by the rule. One of 16 available counters can be used to count packets
Every - match every **Every+1**th packet. For example, if **Every=1** then the rule matches every 2nd packet
Counter - specifies which counter to use. A counter increments each time the rule containing **nth** match matches
Packet - match on the given packet number. The value by obvious reasons must be between **0** and **Every**. If this option is used for a given counter, then there must be at least **Every+1** rules with this option, covering all values between **0** and **Every** inclusively.
out-interface (*name*) - interface the packet is leaving the router through
packet-mark (*text*) - matches packets marked via mangle facility with particular packet mark

packet-size (*integer*: 0..65535-*integer*: 0..65535{0,1}) - matches packet of the specified size or size range in bytes
Min - specifies lower boundary of the size range or a standalone value
Max - specifies upper boundary of the size range
phys-in-interface (*name*) - matches the bridge port physical input device added to a bridge device. It is only useful if the packet has arrived through the bridge
phys-out-interface (*name*) - matches the bridge port physical output device added to a bridge device. It is only useful if the packet will leave the router through the bridge
protocol (ddp | egp | encap | ggp | gre | hmp | icmp | idrp-cmtp | igmp | ipencap | ipip | ipsec-ah | ipsec-esp | iso-tp4 | ospf | pup | rdp | rspf | st | tcp | udp | vmtp | xns-idp | xtp | *integer*) - matches particular IP protocol specified by protocol name or number. You should specify this setting if you want to specify ports
psd (*integer*,*time*,*integer*,*integer*) - attempts to detect TCP and UDP scans. It is advised to assign lower weight to ports with high numbers to reduce the frequency of false positives, such as from passive mode FTP transfers
WeightThreshold - total weight of the latest TCP/UDP packets with different destination ports coming from the same host to be treated as port scan sequence
DelayThreshold - delay for the packets with different destination ports coming from the same host to be treated as possible port scan subsequence
LowPortWeight - weight of the packets with privileged (<=1024) destination port
HighPortWeight - weight of the packet with non-priviliged destination port
random (*integer*) - match packets randomly with given propability
routing-mark (*name*) - matches packets marked by mangle facility with particular routing mark
same-not-by-dst (yes | no) - specifies whether to account or not to account for destination IP address when selecting a new source IP address for packets matched by rules with **action=same**
src-address (*IP address*/*netmask* | *IP address-IP address*) - specifies the address range an IP packet is originated from. Note that console converts entered **address/netmask** value to a valid network address, i.e.:**1.1.1.1/24** is converted to **1.1.1.0/24**
src-address-list (*name*) - matches source address of a packet against user-defined address list
src-address-type (unicast | local | broadcast | multicast) - matches source address type of the IP packet, one of the:
unicast - IP addresses used for one point to another point transmission. There is only one sender and one receiver in this case
local - matches addresses assigned to router's interfaces
broadcast - the IP packet is sent from one point to all other points in the IP subnetwork
multicast - this type of IP addressing is responsible for transmission from one or more points to a set of other points
src-mac-address (*MAC address*) - source MAC address
src-port (*integer*: 0..65535-*integer*: 0..65535{*}) - source port number or range
tcp-mss (*integer*: 0..65535) - matches TCP MSS value of an IP packet
time (*time-time*,sat | fri | thu | wed | tue | mon | sun{+}) - allows to create filter based on the packets' arrival time and date or, for locally generated packets, departure time and date
to-addresses (*IP address-IP address*{0,1}; default: **0.0.0.0**) - address or address range to replace original address of an IP packet with
to-ports (*integer*: 0..65535-*integer*: 0..65535{0,1}) - port or port range to replace original port of an IP packet with

tos (max-reliability | max-throughput | min-cost | min-delay | normal) - specifies a match to the value of Type of Service (ToS) field of IP header
max-reliability - maximize reliability (ToS=4)
max-throughput - maximize throughput (ToS=8)
min-cost - minimize monetary cost (ToS=2)
min-delay - minimize delay (ToS=16)
normal - normal service (ToS=0)

NAT Applications

Description

In this section some NAT applications and examples of them are discussed.

Basic NAT configuration

Assume we want to create router that:

- "hides" the private LAN "behind" one address
- provides Public IP to the Local server
- creates 1:1 mapping of network addresses

Example of Source NAT (Masquerading)

If you want to "hide" the private LAN 192.168.0.0/24 "behind" one address 10.5.8.109 given to you by the ISP, you should use the source network address translation (masquerading) feature of the Hautespot router. The masquerading will change the source IP address and port of the packets originated from the network 192.168.0.0/24 to the address 10.5.8.109 of the router when the packet is routed through it.

To use masquerading, a source NAT rule with action 'masquerade' should be added to the firewall configuration:

```
/ip firewall nat add chain=srcnat action=masquerade out-interface=Public
```

All outgoing connections from the network 192.168.0.0/24 will have source address 10.5.8.109 of the router and source port above 1024. No access from the Internet will be possible to the Local addresses. If you want to allow connections to the server on the local network, you should use destination Network Address Translation (NAT).

Example of Destination NAT

If you want to link Public IP 10.5.8.200 address to Local one 192.168.0.109, you should use destination address translation feature of the Hautespot router. Also if

you want allow Local server to talk with outside with given Public IP you should use source address translation, too

Add Public IP to Public interface:

```
/ip address add address=10.5.8.200/32 interface=Public
```

Add rule allowing access to the internal server from external networks:

```
/ip firewall nat add chain=dstnat dst-address=10.5.8.200 action=dst-nat \
       to-addresses=192.168.0.109
```

Add rule allowing the internal server to talk to the outer networks having its source address translated to 10.5.8.200:

```
/ip firewall nat add chain=srcnat src-address=192.168.0.109 action=src-nat \
       to-addresses=10.5.8.200
```

Example of 1:1 mapping

If you want to link Public IP subnet 11.11.11.0/24 to local one 2.2.2.0/24, you should use destination address translation and source address translation features with **action=netmap**.

```
/ip firewall nat add chain=dstnat dst-address=11.11.11.1-11.11.11.254 \
       action=netmap to-addresses=2.2.2.1-2.2.2.254

/ip firewall nat add chain=srcnat src-address=2.2.2.1-2.2.2.254 \
       action=netmap to-addresses=11.11.11.1-11.11.11.254
```

HauteRouterOS NDP

Document revision: 1.4 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

The Hautespot Neighbor Discovery Protocol (NDP) eases network configuration and management by enabling each Hautespot router to discover other connected Hautespot routers and learn information about the system along with features which are enabled. The Hautespot routers can automatically use learned information to set up some features with minimal or no configuration.

NDP features:

- works on IP level connections
- works on all non-dynamic interfaces
- distributes basic information on the software version
- distributes information on configured features that should interoperate with other Hautespot routers

Hautespot RouterOS is able to discover both NDP and CDP (Cisco Discovery Protocol) devices.

Specifications

Packages required: ***system***
License required: *Level1*
Submenu level: ***/ip neighbor***
Standards and Technologies: MNDP
Hardware usage: *Not significant*

Description

NDP basic function is to assist with automatic configuration of features that are only available between Hautespot routers. Currently this is used for the 'Packet Packer' feature. The 'Packet Packer' may be enabled on a per interface basis. The NDP protocol will then keep information about what routers have enabled the 'unpack' feature and the 'Packet Packer' will be used for traffic between these routers.

Specific features

- works on interfaces that support IP protocol and have at least one IP address and on all ethernet-like interfaces even without IP addresses
- is enabled by default for all new Ethernet-like interfaces -- Ethernet, wireless, EoIP, IPIP tunnels, PPTP-static-server
- when older versions on the RouterOS are upgraded from a version without discovery to a version with discovery, current Ethernet like interfaces will not be automatically enabled for MNDP
- uses UDP protocol port 5678
- a UDP packet with router info is broadcasted over the interface every 60 seconds

- every 30 seconds, the router checks if some of the neighbor entries are not stale
- if no info is received from a neighbor for more than 180 seconds the neighbor information is discarded

Setup

Submenu level: ***/ip neighbor discovery***

Property Description

name (*read-only: name*) - interface name for reference
discover (yes | no; default: **yes**) - specifies whether the neighbor discovery is enabled or not

Example

To disable NDP protocol on Public interface:

```
[admin@Hautespot] ip neighbor discovery> set Public discover=no
[admin@Hautespot] ip neighbor discovery> print
  # NAME      DISCOVER
  0 Public    no
  1 Local     yes
```

Neighbour List

Submenu level: ***/ip neighbor***

Description

This submenu allows you to see the list of neighbors discovered

Property Description

interface (*read-only: name*) - local interface name the neighbor is connected to
address (*read-only: IP address*) - IP address of the neighbor router
mac-address (*read-only: MAC address*) - MAC address of the neighbour router
identity (*read-only: text*) - identity of the neighbor router
version (*read-only: text*) - operating system or firmware version of the neighbor router
unpack (*read-only:* none | simple | compress-headers | compress-all) - identifies if the interface of the neighbor router is unpacking packets packed with M3P
platform (*read-only: text*) - hardware/software platform type of neighbor router
age (*read-only: time*) - specifies the record's age in seconds (time from last update)

Example

To view the table of discovered neighbours:

```
[admin@Hautespot] ip neighbor> pri
 # INTERFACE ADDRESS         MAC-ADDRESS       IDENTITY   VERSION
 0 ether2    10.1.0.113      00:0C:42:00:02:06 ID         2.9beta5
 1 ether2    1.1.1.3         00:0C:42:03:02:ED Hautespot   2.9beta5
[admin@Hautespot] ip neighbor>
```

HauteRouterOS Netwatch Network Monitor

Document revision: 1 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

The Netwatch tool monitors network host by means of ping and generates events on status change.

Specifications

Packages required: ***system***
License required: *Level1*
Submenu level: ***/tool netwatch***
Standards and Technologies: None
Hardware usage: *Not significant*

Network Watching Tool

Specifications

Packages required: ***advanced-tools***
License required: *Level1*
Submenu level: ***/tool netwatch***
Standards and Technologies: none
Hardware usage: *Not significant*

Description

Netwatch monitors state of hosts on the network. It does so by sending ICMP pings to the list of specified IP addresses. For each entry in netwatch table you can specify IP address, ping interval and console scripts. The main advantage of netwatch is it's ability to issue arbitrary console commands on host state changes.

Property Description

down-script (*name*) - a console script that is executed once when state of a host changes from **unknown** or **up** to **down**
host (*IP address*; default: **0.0.0.0**) - IP address of host that should be monitored
interval (*time*; default: **1s**) - the time between pings. Lowering this will make state changes more responsive, but can create unnecessary traffic and consume system resources
since (*read-only: time*) - indicates when state of the host changed last time
status (*read-only:* up | down | unknown) - shows the current status of the host
up - the host is up
down - the host is down
unknown - after any properties of this list entry were changed, or the item is enabled or disabled
timeout (*time*; default: **1s**) - timeout for each ping. If no reply from a host is received during this time, the host is considered unreachable (**down**)
up-script (*name*) - a console script that is executed once when state of a host changes from **unknown** or **down** to **up**

Example

This example will run the scripts gw_1 or gw_2 which change the default gateway depending on the status of one of the gateways:

```
[admin@Hautespot] system script> add name=gw_1 source={/ip route set
{... [/ip route find dst 0.0.0.0] gateway 10.0.0.1}
[admin@Hautespot] system script> add name=gw_2 source={/ip route set
{.. [/ip route find dst 0.0.0.0] gateway 10.0.0.217}
[admin@Hautespot] system script> /tool netwatch
[admin@Hautespot] tool netwatch> add host=10.0.0.217 interval=10s
timeout=998ms \
\... up-script=gw_2 down-script=gw_1
[admin@Hautespot] tool netwatch> print
Flags: X - disabled
  #   HOST            TIMEOUT               INTERVAL              STATUS
  0   10.0.0.217      997ms                 10s                   up
[admin@Hautespot] tool netwatch> print detail
Flags: X - disabled
  0   host=10.0.0.217 timeout=997ms interval=10s since=feb/27/2003 14:01:03
      status=up up-script=gw_2 down-script=gw_1

[admin@Hautespot] tool netwatch>
```

Without scripts, netwatch can be used just as an information tool to see which links are up, or which specific hosts are running at the moment.

Let's look at the example above - it changes default route if gateway becomes unreachable. How it's done? There are two scripts. The script "gw_2" is executed once when status of host changes to **up**. In our case, it's equivalent to entering this console command:

```
[admin@Hautespot] > /ip route set [/ip route find dst 0.0.0.0] gateway
10.0.0.217
```

The **/ip route find dst 0.0.0.0** command returns list of all routes whose **dst-address** value is **0.0.0.0**. Usually, that is the default route. It is substituted as first argument to **/ip route set** command, which changes gateway of this route to 10.0.0.217

The script "gw_1" is executed once when status of host becomes **down**. It does the following:

```
[admin@Hautespot] > /ip route set [/ip route find dst 0.0.0.0] gateway
10.0.0.1
```

It changes the default gateway if 10.0.0.217 address has become unreachable.

Here is another example, that sends e-mail notification whenever the 10.0.0.215 host goes down:

```
[admin@Hautespot] system script> add name=e-down source={/tool e-mail send
{... from="rieks@mt.lv" server="159.148.147.198" body="Router down"
{... subject="Router at second floor is down" to="rieks@latnet.lv"}
[admin@Hautespot] system script> add name=e-up source={/tool e-mail send
{... from="rieks@mt.lv" server="159.148.147.198" body="Router up"
{.. subject="Router at second floor is up" to="rieks@latnet.lv"}
[admin@Hautespot] system script>
[admin@Hautespot] system script> /tool netwatch
[admin@Hautespot] system netwatch> add host=10.0.0.215 timeout=999ms \
\... interval=20s up-script=e-up down-script=e-down
[admin@Hautespot] tool netwatch> print detail
Flags: X - disabled
  0   host=10.0.0.215 timeout=998ms interval=20s since=feb/27/2003 14:15:36
      status=up up-script=e-up down-script=e-down

[admin@Hautespot] tool netwatch>
```

HauteRouterOS Mesh Routing Protocol

Document revision: 2.2 (Tuesday, October 30, 2007)

Applies to: HauteRouterOS V3.0

Overview

MME (Mesh Made Easy) is a HauteSpot routing protocol suited for IP level routing in wireless mesh networks. It is based on ideas from B.A.T.M.A.N. (Better Approach To Mobile Ad-hoc Networking) routing protocol. See https://www.open-mesh.net for more information about B.A.T.M.A.N.

MME works by periodically broadcasting so called originator messages. Routing information contained in a message consists of IP address of it's originator and optional list of IP prefixes - network announcements. If a node receives an originator message it hasn't seen before, it rebroadcasts that message. (There also are some other cases when the message can be rebroadcasted - see below.)

Unlike OLSR or other "traditional" proactive routing protocols, MME does not maintain network topology information. Consequently, MME is not able to calculate routing table, and does not need to. Instead, it keeps tracks of packets received and their sequence numbers - to tell how many packets were lost. This way, from message loss statistics for all combinations of originators and single-hop neighbors, MME is able to find the best gateway to a particular destination.

The main ideas behind MME are based on these observations made in mobile mesh networks:

- it can be impossible to know the exact topology of all network, because it is rapidly changing;
- if topology changes trigger routing table recalculation for all nodes in the network; and for embedded systems, the routing table calculation CPU overhead can be significant.

To avoid these problems, a MME node:

- cares only about the best single-hop neighbor in path to a particular destination;
- avoids routing table calculations.

Secondary functions of the MME protocol are: to carry information about gateways to the Internet, and to dynamically setup default routes. The part of MME responsible for that is dubbed "the gateway protocol".

MME protocol is using UDP port 1966 for originator message traffic. The gateway protocol is using TCP port 1968.

It is assumed in a normal operation of the protocol, a large number of these messages will get lost due to bad link quality. This assumption is important if we are talking about protocol overhead. Theoretically protocol's own traffic consumption is at least as big as for RIP, and obviously worse than that of link state routing protocols (OSPF, OLSR) unless the topology is constantly changing.

Technical side

Basic principles of the main protocol

The main functions of the MME protocol are:

- automatic neighbor MME router (so called "originator") discovery (including multihop neighbors);
- originator message origination and flooding on each interface in every **origination-interval** seconds;
- originator message rebroadcasting based on a few simple rules;
- best gateway selection for each originator and the routes it has advertised.

Originator message rebroadcasting rules:

- do not rebroadcast self originated messages;
- do not rebroadcast messages that has *unidirectional* flag set;
- rebroadcast messages from single-hop neighbors; rebroadcast with *unidirectional* flag set if and only if:
    - the neighbor relation is not bidirectional;
    - OR the neighbor is not the best gateway to himself (i.e. there exists a better multihop path towards this node).
- rebroadcast messages that are not duplicate; a message is considered duplicate if message with this sequence number already was received before;
- rebroadcast duplicate messages if and only if:
    - they came from a neighbor that is the gateway for the originator;
    - the TTL in the packet is equal to last TTL for this neighbor and originator combination.

MME makes routing decisions based no more than last 64 messages received, but this number can be significantly less in case of packet loss. The node can tell that some packets were lost based on their sequence numbers. The more originator messages are received from a node, the better the statistics of that node is.

The MME protocol does not incorporate best route selection logic. If the same network information is configured in two different nodes, there currently is no way how to tell which one to prefer. Both routes will be installed in routing table and one of the selected in a random fashion. Obviously, such configuration is not recommended.

Basic principles of the gateway protocol

Second part of the MME is a default gateway selection protocol. Here two roles for a router are possible. A *gateway server* is node that is willing to serve as internet gateway for other routers. Usually it means it has an Ethernet connection or some other way "out of the mesh".

A *gateway client* is a node that is willing to use this dynamic information to about gateways out of the mesh cloud. If there are multiple gateways reachable, client selects the best one based on packet statistics, advertised gateway class, and

gateway-selection and **preferred-gateway** configuration values. After selecting the best gateway server the client makes a TCP connection to the server. This connection is used for periodic keep-alive message sending. After the connection is established, both the client and the server add dynamic IPIP tunnel interface. The client also adds default route through this interface.

If the server stops announcing it's gateway capability, or becomes unreachable, the TCP connection and all tunnel state is torn down on both sides. Client also removes the default route.

Note that it's not recommended to have a default route (i.e. prefix 0.0.0.0/0) in MME network announcement configuration.

Packet format

The one and only packet type used in MME is *originator message*. The message contains:

- originator IP;
- current ttl value;
- sequence number;
- gateway class;
- protocol version;
- host and network announcements (0..n IP prefixes).

Gateway protocol clients and servers also exchange *keep-alive messages*, but they contain no information and have undefined format. At the moment, however, a keep-alive message is considered invalid, it if contains fewer than 1 or more than 6 octets.

Configuration examples

Starting the protocol on a single interface:

```
[admin@I] > /routing mme interface add interface=wlan1
```

To change some attributes for routes learned via MME you can use the **mme-in** routing filter. Example:

```
[admin@HauteSpot] > routing filter add chain=mme-in set-routing-mark=mark1
```

If you want to redistribute some routes via MME, add them to MME networks. Example:

```
[admin@HauteSpot] /routing mme> network add network=1.2.3.0/24
[admin@HauteSpot] /routing mme> network p
Flags: X - disabled
 #   NETWORK
 0   1.2.3.0/24
```

Using the gateway protocol

Setup gateway server:

```
[admin@I] /routing mme> set gateway-class=11
```

Setup gateway client:

```
[admin@HauteSpot] /routing mme> set gateway-selection=best-statistic
```

Observe the results (on client). Dynamic IPIP interface should be added automatically:

```
[admin@HauteSpot] > /interface print
Flags: X - disabled, D - dynamic, R - running
 #    NAME                                TYPE          MTU
 0  R ether1                              ether         1500
 1  R ether2                              ether         1500
 2 DR ipip1                               ipip          1480
```

Default route that goes through this tunnel should be added automatically:

```
[admin@HauteSpot] > /ip route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        G GATEWAY        DISTANCE INTERFACE
 0 ADm  0.0.0.0/0                          r ipip1          130      ipip1
```

MME command reference

Specifications

Packages required: **routing**
License required: Level1
Submenu level: **/routing mme**

General Setup

Property Description

origination-interval (*time*; default: **5s**) - interval between originator messages. Obviously, this value should be less than **timeout** value.

timeout (*time*; default: **1m**) - node timeout. If no messages at all are received from an originator node during this interval, that node is purged from protocol tables, and so are all routes it has announced.

bidirectional-timeout (*integer*; default: **2**) - how many originator messages from a node can be lost in sequence, while still considering it a bidirectional neighbor. We are assuming that every node originates messages with the same rate as this router (i.e. the value from **origination-interval**).

ttl (*integer*; default: **50**) - how many times to forward originator messages.

gateway-class (*none* | *56-KBit* | *64-KBit* | *128-KBit* | *256-KBit* | *512-KBit* | *1-MBit* | *2-MBit* | *3-MBit* | *5-MBit* | *6-MBit* | *>6-MBit* | *integer*; default: **none**) - announce internet gateway capability in the originator messages sent by this node.

The node running MME with gateway-class option is supposed to have a link to Internet and a default route to that. The symbolic values are compatible with B.A.T.M.A.N. This table describes the mapping from integers to symbolic values:

```
0 no gateway
1 modem
2 ISDN
3 Double ISDN
3 256 KBit
5 UMTS/ 0.5 MBit
6 1 MBit
7 2 MBit
8 3 MBit
9 5 MBit
10 6 MBit
11 >6 MBit
```

Entering integer value > 11 means even better gateway class.
gateway-selection (*no-gateway* | *best-adjusted* | *best-statistic*; default: **no-gateway**) - this node is a MME gateway protocol client.

- **no-gateway** - don't install default route via MME.
- **best-adjusted** - select best gateway node based on received message statistics and announced gateway class;
- **best-statistic** - select best gateway node based only on received message statistics;

gateway-keepalive (*time*; default: **1m**) - the time interval between successive gateway keepalive messages. For gateway client, this specifies how often to send out keepalive messages. For gateway server, as client hold time is used 3 * **gateway-keepalive** seconds. If the server does not receive keepalive messages from a client during this time interval, the client is considered dead. All state information associated with it are deleted, including the dynamic IPIP tunnel.
preferred-gateway (*IP address*; default: **0.0.0.0**) - always prefer this node as internet gateway to any others, if it is present in originator tables.

Interfaces

Submenu level: **/routing mme interface**
Description: List of interfaces on which to run the MME protocol.

interface (*name*; default: **all**) - interface on which MME will run

- **all** - is used for the interfaces not having any specific settings

passive (*yes* | *no*; default: **no**) - if true, do not send originator messages via this interface, only receive.

primary (*yes* | *no*; default: **no**) - include routing information (i.e. network announcements) in self-originated packets send via this interface. (For forwared packets the information is always included.) Only one interface can be primary. If no interfaces are configured as primary, one is selected automatically in a random fashion.

messages-tx (*read only: integer*) - originator messages transmitted via this interface. For **all** interface: cumulative statistics.

messages-rx (*read only: integer*) - originator messages received via this interface. For **all** interface: cumulative statistics.

Networks

Submenu level: **/routing mme network**
Description: MME Networks is a list of networks to be advertised.

network (*IP prefix*) - network to advertise

Notes

The usage of MME networks is similar to BGP networks, and different from IGP (i.e. RIP and OSPF) networks. They determine which networks to announce via MME, not on which networks to run the protocol.

Status

Submenu level: **/routing mme status**

Description: Read-only status information about the protocol. Currently has only one submenu: **originators** - information about active neighbor nodes.

originator (*read-only: IP address*) - the IP address of the node.

gateway (*read-only: IP address*) - the nexthop for this node.

gateway-class (*read-only: none* | *56-KBit* | *64-KBit* | *128-KBit* | *256-KBit* | *512-KBit* | *1-MBit* | *2-MBit* | *3-MBit* | *5-MBit* | *6-MBit* | *>6-MBit* | *integer*) - if **none**, then this node is not a gateway server. Otherwise this node is not a gateway server with specified gateway bandwidth.

last-packet-before (*read-only: time*) - seconds elapsed since last received packet.

Optimizing HR-WRAP for Video

The HR-WRAP device has the ability to use a combination of proprietary and non-proprietary features to ensure maximum stability when transmitting video data. Video data is

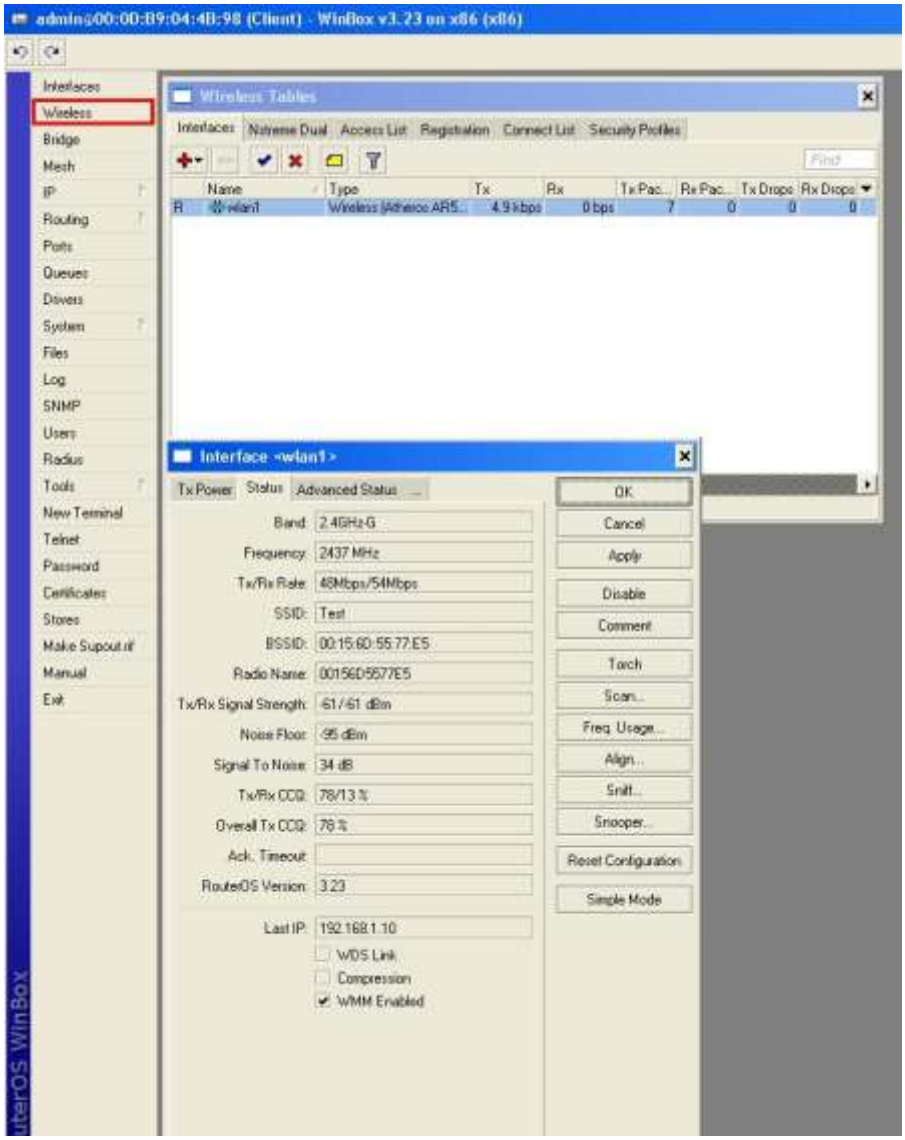
different from other IP data because it relies on a steady low latency link to ensure smooth synchronized video. When implementing a video network it is important to consider two other key concepts. Video **encapsulation protocol** and **transportation protocol** will greatly affect your videos ability to be transmitted wirelessly. By now these have probably already been chosen but it is important to know and understand each for optimization.

Video **encapsulation protocol** is usually very well documented. Common encapsulation protocols are MPEG4, MPEG4H.264 and Motion JPEG. We encourage the use of MPEG4H.264 for the best resolution, frame rate and stability.

Video **transportation protocol** is not always considered by some network installers however can make or break a network. HTTP, RTSP, RTP, Unicast and Multicast and MPEGTS are all very common in the camera/encoder markets. An unacknowledged protocol such as RTSP, RTP, Unicast or Multicast will ensure maximum stability/recovery of the network. When using an acknowledged protocol the receiver has to acknowledge every video packet received by the transmitter. If a piece of data is not acknowledged as being received the whole packet/acknowledgement sequence must start again. This is extremely difficult for video since it relies on a low latency sequential data stream.

Optimization should occur after configuration and installation however links may be broken during the optimization process so direct access may be necessary. Optimization does not include hardware tuning such as antenna alignment and cable sealing.

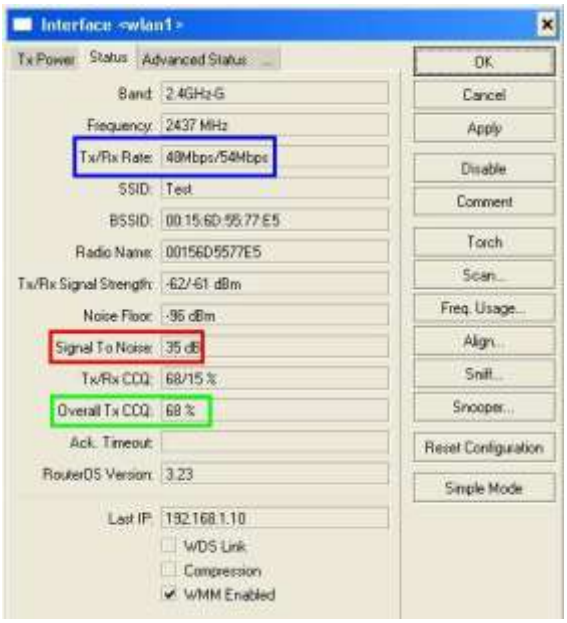
By default our WMM and Nstreme features have been enabled. This ensures sequential and prioritized video data even when using multiple cameras linked to a single access point. O*ptimization is going to involve adjusting data rates to achieve 100% data transportation while maximizing data rate*. Data rates are a fairly simple concept. A higher data rate (54mbps) will use less error detection to ensure higher bandwidth. This is poor for a low quality link because their will be lots of undetected errors. A lower data rate (6mbps) uses lots of error detection at the expense of bandwidth. Our link quality level will be based on a value called **CCQ** (client connect quality) and bandwidth stability.



Optimization will occur on the client radios. A optimized configuration is dependent on link quality and will need to be done for every client. Start with your most physically accessible client since broken links/direct access is usually required when learning.

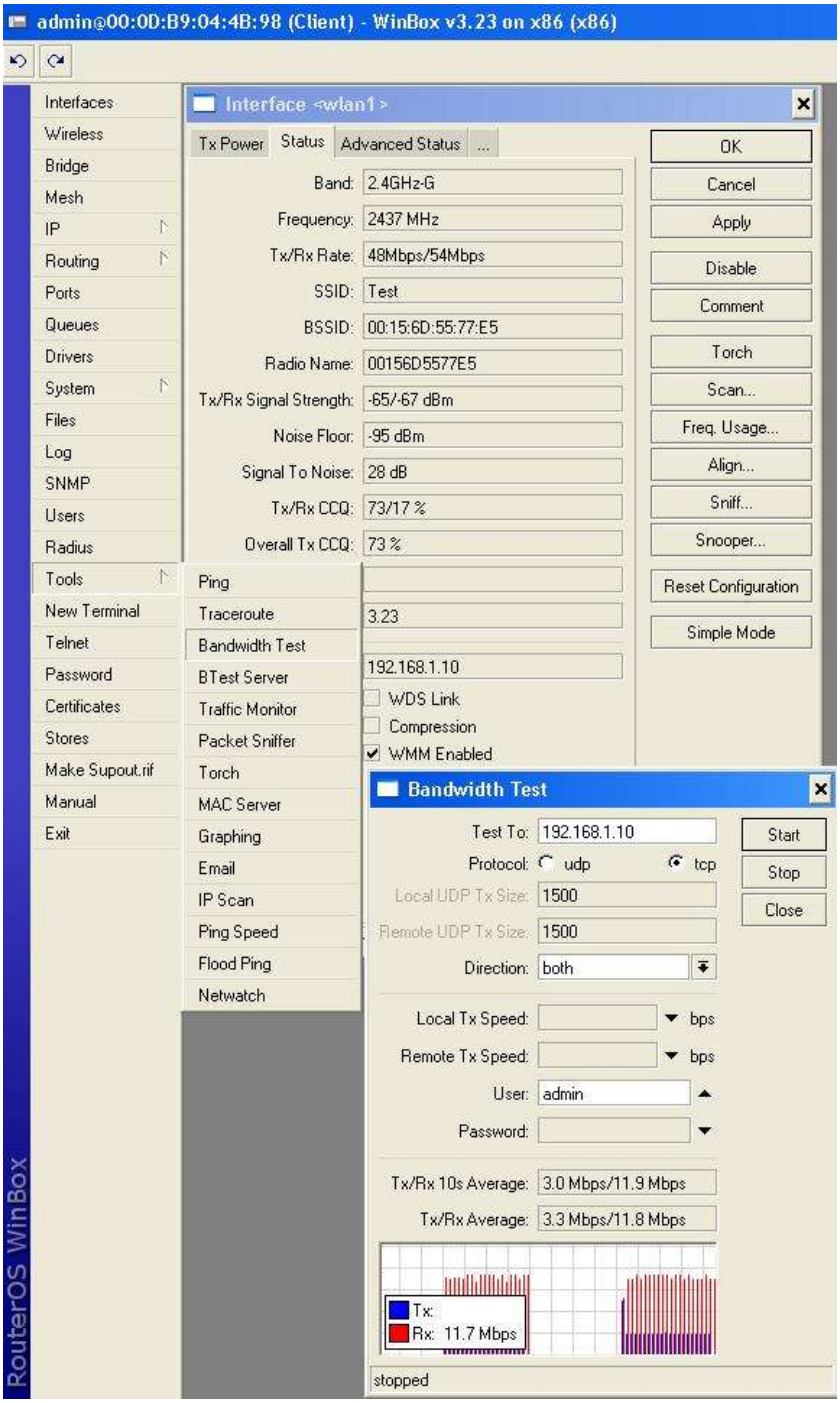
The Wireless Tables menu is where most of our configuration changes will be made. To open it, select the Wireless button highlighted in the figure above. Double click wlan1 to open the Interface <wlan 1> menu.

Once the Interface <wlan1> menu has been opened ensure you are operating in 'Advanced Mode'. Simply click the 'Advanced Mode' button at the bottom right of the window. When in Advanced Mode the button will toggle to 'Simple Mode' like pictured above. Next, navigate to the 'Status' tab in the Interface <wlan1> window. To move tabs you can either select the '…' button or just keep clicking the right/left most tab to walk from tab to tab.



Once in the Status tab you will need to generate data across the wireless link. Most of information in the status tab is an average over time. The more data being averaged the more accurate your information will be. Data may be generated lots of sources. Ideally however we encourage the use of the provided bandwidth test tool. This not only generates data but provides a benchmark for your wireless bandwidth.

To open the bandwidth test tool select the 'Tools' button. In the sub menu select 'Bandwidth Test'.

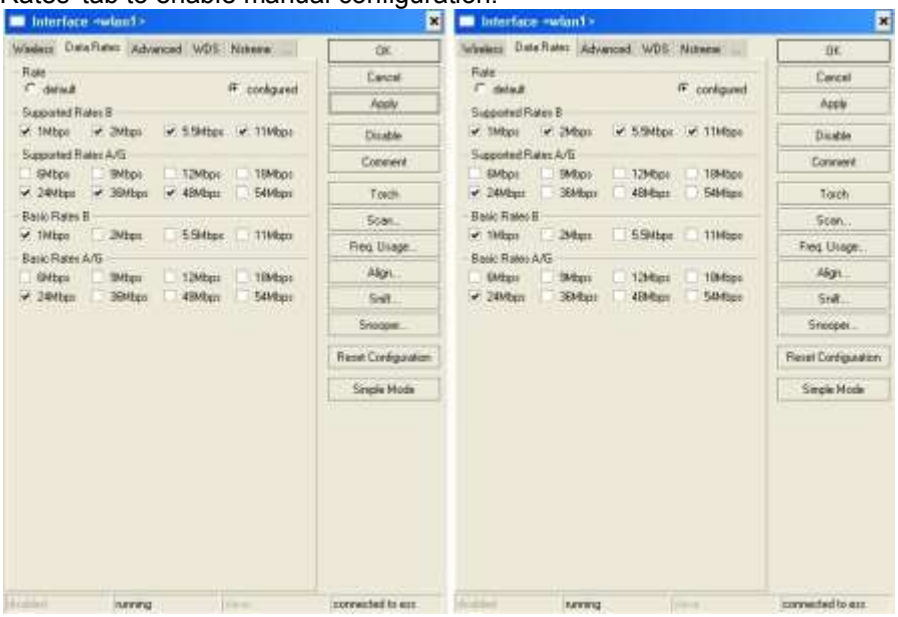


To start the bandwidth test you will need to enter the IP address of your access point device as well as the login information (admin for username by default). We suggest using TCP with a direction of both for the most demanding test. Once the test is started you will be given instant bandwidth, average bandwidth over 10 seconds and total average bandwidth. Notice the instant Tx bandwidth above is blank. This is normal and should recover shortly. Also do not run the bandwidth tool in conjunction with streaming video, results will suffer.

While running the bandwidth test bring up your 'Interface <wlan1>' menu and look at the 'Status' tab. Now that your wireless link is loaded with data your status information should be current and accurate. At this point look take a look at your Signal to Noise. If below 25dB we encourage finding a lower noise floor or strengthening your link. This may be done by cleaning and sealing pigtails, aligning antennas, changing frequencies etc.

Next check out your Overall CCQ and Tx/Rx Rate. When your device is auto adjusting data rates (default behavior) it should find a data rate that provides a fairly decent CCQ. Typically a device will bounce between data rates that provide 50%-100% CCQ. You can see when the device jumps to the higher data rate your CCQ percentage will drop. When the device recognizes that there are too many errors at that data rate the data rate will drop and your CCQ percentage will go up.

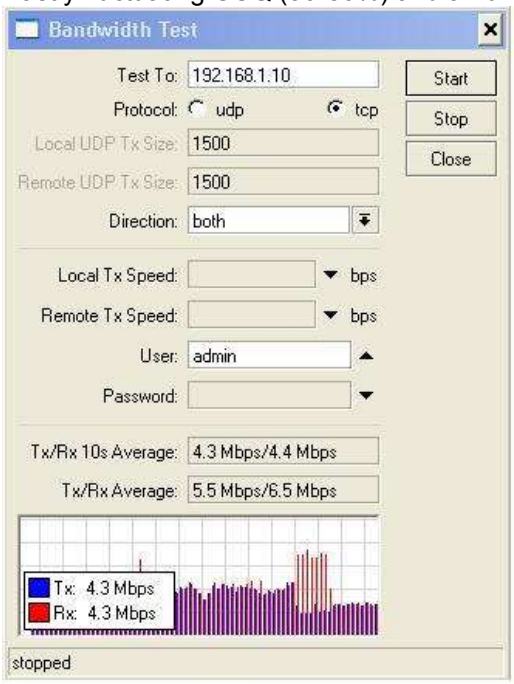
This is where tuning begins. Our job is to eliminate the bouncing behavior as well as ensure a good CCQ/Bandwidth. This is achieved by fixing the data rate (instead of the automatic adjustment) to a data rate that provides a high CCQ. This requires that we navigate to the 'Data Rates' tab in the 'Interface <wlan1>' menu. This tab won't exist if you are not in Advanced Mode. Notice we also must have the 'configured' radio button selected in the 'Data Rates' tab to enable manual configuration.



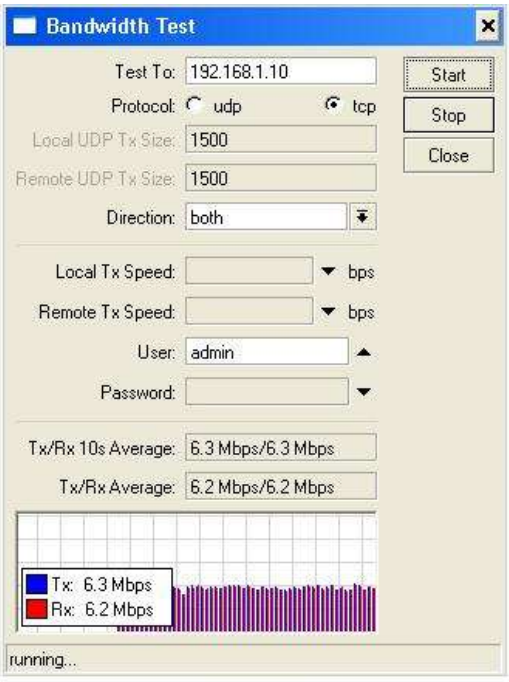
The data rates configuration can be confusing. Data rates are broken up into two modes. A/G and B. We will not be making any changes to the B rates. A basic rate is the starting data rate. There is only one basic rate. This is the rate that the link will be beginning at. The supported rates are the rates that the device is allowed to move *up* to. A supported rate can not be below the basic rate. In the figure above the left configuration is configured to start at the 24Mbps with the ability to move up to 48Mbps. In the configuration on the right the device is stuck at the 24Mbps rate. We will be dealing mostly with a configuration like the one on the right (one fixed data rate).

Configure your device to the data rate that you found appropriate when watching the 'Status' tab. Select 'Apply'. This is the point at which your link will either break or become better. Assuming that you are logged into the client device remotely (over the wireless link) a broken link will not be obvious. Bring up your Bandwidth Test to ensure that the results are updating. If the test is still running and the results are updating you know you have not killed your link. If the test has stopped and can not be restarted chances are the link has died. If this is the case you will need to get a physical Ethernet connection into the remote device so that the changes can be un-done. While on location the user can continue tuning without the frustration of the wireless link breaking.

Once the data rate has been fixed and the link established (keep your bandwidth test running) navigate back to the 'Status' tab. You can now see that your 'Tx/Rx Rate' is fixed at data rate you mandated. You can also see that your CCQ is a lot more steady. A stable link will have a CCQ that is above 70% with a stable bandwidth test result. A bad link will have a vastly fluctuating CCQ (30-80%) and an unstable bandwidth test result.



Bad



Good

The two bandwidth test's above were between the exact same devices using a different fixed data rate. The one on the left was using a fixed data rate of 54Mbps whereas the one on the right was using a fixed data rate of 24Mbps.

If your fixed data rate does not provide a stable bandwidth test then move down one data rate and repeat the bandwidth test. If this still provides a fluctuating bandwidth test then you may want to configure the device for default data rates (automatic adjustment) and see where it stabilizes itself. The automatic data rate adjustment will always put itself in the ballpark range ideal for video.

Every client may use a different data rate (depending on link quality). You will have to optimize the data rate independently for every client on the network.

OSPF to simulate full duplex links with redundancy

Router #1 Local

Enable both wlan connections, set one up as an wlan1 ap and wlan2 as a station

```
[admin@HauteSpot] /interface wireless> print
```

Flags: X - disabled, R - running

```
0  R name="wlan1" mtu=1500 mac-address=00:0C:42:05:64:F3 arp=enabled
interface-type=Atheros AR5413 mode=ap-bridge ssid="hautespot"
     frequency=5180 band=5ghz scan-list=default antenna-mode=ant-a wds-
mode=disabled wds-default-bridge=none wds-ignore-ssid=no
     default-authentication=yes default-forwarding=yes default-ap-tx-
limit=0 default-client-tx-limit=0 hide-ssid=no
     security-profile=default compression=no
1  R name="wlan2" mtu=1500 mac-address=00:0C:42:1B:0B:7D arp=enabled
interface-type=Atheros AR5413 mode=station ssid="hautespot2"
     frequency=5220 band=5ghz scan-list=default antenna-mode=ant-a wds-
mode=disabled wds-default-bridge=none wds-ignore-ssid=no
     default-authentication=yes default-forwarding=yes default-ap-tx-
limit=0 default-client-tx-limit=0 hide-ssid=no
     security-profile=default compression=no
```

set ether1 to any Local lan address 192.168.209.137/24 set wlan1 to a router subnet address 192.168.88.131/30 -- I use small subnets for p2p links for efficiency set wlan2 to a router address on a different subnet 192.168.88.132/30

[admin@HauteSpot] /ip address> print Flags: X - disabled, I - invalid, D - dynamic

```
#   ADDRESS            NETWORK         BROADCAST       INTERFACE
0   192.168.88.129/30  192.168.88.128  192.168.88.131  wlan1
1   192.168.88.134/30  192.168.88.132  192.168.88.135  wlan2
2  192.168.209.137/24 192.168.209.0   192.168.209.255 ether1
```

Create an OSPF area like 0.0.0.1 named something like remote-net

[admin@ HauteSpot] /routing ospf area> print Flags: X - disabled

```
#   NAME                  AREA-ID         TYPE    DEFAULT-COST
0   backbone              0.0.0.0         default
1   remote-net            0.0.0.1         default
```

add your remote network and routed links to the area (mine is all in one /24 even though it is three subnets)

[admin@ HauteSpot] /routing ospf network> print Flags: X - disabled, I - invalid

```
#   NETWORK            AREA
0   192.168.88.0/24    remote-net
```

Your interfaces will automatically populate when you add the second router.....

Router#2 Remote Enable both wlan connections, set one up as an wlan1 ap and wlan2 as a station

[admin@ HauteSpot] /interface wireless> print Flags: X - disabled, R - running

```
0  R name="wlan1" mtu=1500 mac-address=00:0C:42:1B:0B:84 arp=enabled
interface-type=Atheros AR5413 mode=ap-bridge
     ssid="2SCTelcom" frequency=5200 band=5ghz scan-list=default
antenna-mode=ant-a wds-mode=disabled wds-default-bridge=none
     wds-ignore-ssid=no default-authentication=yes default-
forwarding=yes default-ap-tx-limit=0 default-client-tx-limit=0
     hide-ssid=no security-profile=default compression=no
1  R name="wlan2" mtu=1500 mac-address=00:0C:42:1B:0B:57 arp=enabled
interface-type=Atheros AR5413 mode=station
     ssid="2Sutton" frequency=5180 band=5ghz scan-list=default antenna-
mode=ant-a wds-mode=disabled wds-default-bridge=none
     wds-ignore-ssid=no default-authentication=yes default-
forwarding=yes default-ap-tx-limit=0 default-client-tx-limit=0
     hide-ssid=no security-profile=default compression=no
```

set ether1 to the gateway address you will be using at the remote network 192.168.88.1/25

set wlan1 to a router address you are receiving from on router #1 192.168.88.130/30

set wlan2 to a router address on a the subnet that you are broadcasting from in the router #1 192.168.88.133/30

[admin@ HauteSpot] /ip address> print Flags: X - disabled, I - invalid, D - dynamic

```
#   ADDRESS            NETWORK         BROADCAST       INTERFACE
0   ;;; default configuration
    192.168.88.1/24    192.168.88.0    192.168.88.255  ether1
1   192.168.88.133/30  192.168.88.132  192.168.88.135  wlan1
2   192.168.88.130/30  192.168.88.128  192.168.88.131  wlan2
```

Create the same OSPF areas and networks you made on router #1

[admin@ HauteSpot] /routing ospf> area print Flags: X - disabled

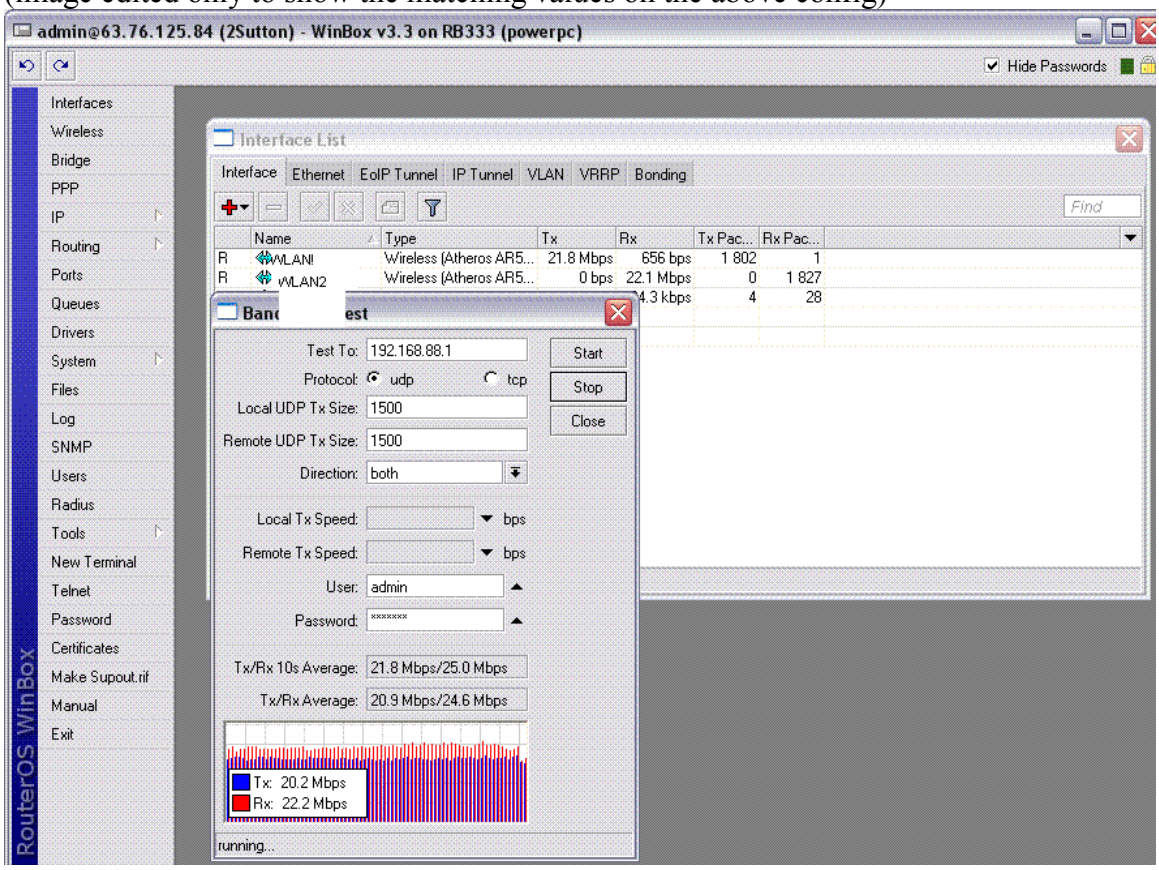
```
#    NAME                    AREA-ID         TYPE     DEFAULT-COST
0    backbone                0.0.0.0         default
1    remote-net              0.0.0.1         default
```

[admin@ HauteSpot] /routing ospf> network print Flags: X - disabled, I - invalid

```
#    NETWORK             AREA
0    192.168.88.0/24     remote-net
```

Now Run a bandwidth test between the routers

(image edited only to show the matching values on the above config)



HauteRouterOS OSPF

Document revision: 1.4 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

HauteRouterOS implements OSPF Version 2 (RFC 2328). The OSPF protocol is the link-state protocol that takes care of the routes in the dynamic network structure that can employ different paths to its subnetworks. It always chooses shortest path to the subnetwork first.

Specifications

Packages required: ***routing***
License required: *Level3*
Submenu level: ***/routing ospf***
Standards and Technologies: OSPF
Hardware usage: *Not significant*

Description

Open Shortest Path First protocol is a link-state routing protocol. It's uses a link-state algorithm to build and calculate the shortest path to all known destinations. The shortest path is calculated using the Dijkstra algorithm. OSPF distributes routing information between the routers belonging to a single autonomous system (AS). An AS is a group of routers exchanging routing information via a common routing protocol.

In order to deploy the OSPF all routers it will be running on should be configured in a coordinated manner (note that it also means that the routers should have the same MTU for all the networks advertized by OSPF protocol).

The OSPF protocol is started after you will add a record to the OSPF network list. The routes learned by the OSPF protocol are installed in the routes table list with the distance of 110.

General Setup

Submenu level: ***/routing ospf***

Description

In this section you will learn how to configure basic **OSPF** settings.

Property Description

distribute-default (never | if-installed-as-type-1 | if-installed-as-type-2 | always-as-type-1 | always-as-type-2; default: **never**) - specifies how to distribute default route. Should be used for ABR (Area Border router) or ASBR (Autonomous System boundary router) settings
never - do not send own default route to other routers
if-installed-as-type-1 - send the default route with type 1 metric only if it has been installed (a static default route, or route added by DHCP, PPP, etc.)
if-installed-as-type-2 - send the default route with type 2 metric only if it has been installed (a static default route, or route added by DHCP, PPP, etc.)
always-as-type-1 - always send the default route with type 1 metric
always-as-type-2 - always send the default route with type 2 metric
metric-bgp (*integer*; default: **20**) - specifies the cost of the routes learned from BGP protocol
metric-connected (*integer*; default: **20**) - specifies the cost of the routes to directly connected networks
metric-default (*integer*; default: **1**) - specifies the cost of the default route
metric-rip (*integer*; default: **20**) - specifies the cost of the routes learned from RIP protocol
metric-static (*integer*; default: **20**) - specifies the cost of the static routes
redistribute-bgp (as-type-1 | as-type-2 | no; default: **no**) - with this setting enabled the router will redistribute the information about all routes learned by the BGP protocol
redistribute-connected (as-type-1 | as-type-2 | no; default: **no**) - if set, the router will redistribute the information about all connected routes, i.e., routes to directly reachable networks
redistribute-rip (as-type-1 | as-type-2 | no; default: **no**) - with this setting enabled the router will redistribute the information about all routes learned by the RIP protocol
redistribute-static (as-type-1 | as-type-2 | no; default: **no**) - if set, the router will redistribute the information about all static routes added to its routing database, i.e., routes that have been created using the **/ip route add command**
router-id (*IP address*; default: **0.0.0.0**) - OSPF Router ID. If not specified, OSPF uses the largest IP address configured on the interfaces as its router ID

Notes

Within one area, only the router that is connected to another area (i.e. Area border router) or to another AS (i.e. Autonomous System boundary router) should have the propagation of the default route enabled.

OSPF protocol will try to use the shortest path (path with the smallest total cost) if available.

OSPF protocol supports two types of metrics:

- **type1** - external metrics are expressed in the same units as OSPF interface cost. In other words the router expects the cost of a link to a network which is external to AS to be the same order of magnitude as the cost of the internal links.
- **type2** - external metrics are an order of magnitude larger; any **type2** metric is considered greater than the cost of any path internal to the AS. Use of **type2** external metric assumes that routing between AS is the major cost of routing a packet, and climinates the need conversion of external costs to internal link state metrics.

Both Type 1 and Type 2 external metrics can be used in the AS at the same time. In that event, Type 1 external metrics always take precedence.

In **/ip route** you can see routes with **Io** status. Because router receives routers from itself.

The metric cost can be calculated from line speed by using the formula 10e+8/line speed. The table contains some examples:

network type	cost
ethernet	10
T1	64
64kb/s	1562

Example

To enable the OSPF protocol redisrtibute routes to the connected networks as **type1** metrics with the cost of **1**, you need do the following:

```
[admin@HauteSpot] routing ospf> set redistribute-connected=as-type-1 \
\... metric-connected=1
[admin@HauteSpot] routing ospf> print
                  router-id: 0.0.0.0
        distribute-default: never
    redistribute-connected: as-type-1
       redistribute-static: no
          redistribute-rip: no
          redistribute-bgp: no
            metric-default: 1
          metric-connected: 1
             metric-static: 20
                metric-rip: 20
                metric-bgp: 20
[admin@HauteSpot] routing ospf>
```

Areas

Submenu level: ***/routing ospf area***

Description

OSPF allows collections of routers to be grouped together. Such group is called an area. Each area runs a separate copy of the basic link-state routing algorithm. This means that each area has its own link-state database and corresponding graph

The structure of an area is invisible from the outside of the area. This isolation of knowledge enables the protocol to effect a marked reduction in routing traffic as compared to treating the entire Autonomous System as a single link-state domain

60-80 routers have to be the maximum in one area

Property Description

area-id (*IP address*; default: **0.0.0.0**) - OSPF area identifier. Default **area-id=0.0.0.0** is the backbone area. The OSPF backbone always contains all area border routers. The backbone is responsible for distributing routing information between non-backbone areas. The backbone must be contiguous. However, areas do not need to be physical connected to backbone. It can be done with virtual link. The name and area-id for this area can not be changed
authetication (none | simple | md5; default: **none**) - specifies authentication method for OSPF protocol messages
none - do not use authentication
simple - plain text authentication
md5 - keyed Message Digest 5 authentication
default-cost (*integer*; default: **1**) - specifies the default cost used for stub areas. Applicable only to area boundary routers
name (*name*; default: **""**) - OSPF area's name
stub (yes | no; default: **no**) - a stub area is an area which is out from part with no routers or areas beyond it. A stub area is configured to avoid AS External Link Advertisements being flooded into the Stub area. One of the reason to configure a Stub area is that the size of the link state database is reduced along with the routing table and less CPU cycles are used to process. Any router which is trying access to a network outside the area sends the packets to the default route

Example

To define additional OSPF area named **local_10** with **area-id=0.0.10.5**, do the following:

```
[admin@WiFi] routing ospf area> add area-id=0.0.10.5 name=local_10
[admin@WiFi] routing ospf area> print
Flags: X - disabled, I - invalid
 #   NAME                    AREA-ID         STUB DEFAULT-COST
AUTHENTICATION
 0   backbone                0.0.0.0                          none
 1   local_10                0.0.10.5        no   1           none
```

```
[admin@WiFi] routing ospf area>
```

Networks

Submenu level: ***/routing ospf network***

Description

There can be Point-to-Point networks or Multi-Access networks. Multi-Access network can be a broadcast network (a single message can be sent to all routers)

To start the OSPF protocol, you have to define the networks on which it will run and the area ID for each of those networks

Property Description

area (*name*; default: **backbone**) - the OSPF area to be associated with the specified address range
network (*IP address mask*; default: **20**) - the network associated with the area. The **network** argument allows defining one or multiple interfaces to be associated with a specific OSPF area. Only directly connected networks of the router may be specified

Notes

You should set the network address exactly the same as the remote point IP address for point-to-point links. The right netmask in this case is **/32**.

Example

To enable the OSPF protocol on the 10.10.1.0/24 network, and include it into the backbone area, do the following:

```
[admin@HauteSpot] routing ospf network> add area=backbone network=10.10.1.0/24
[admin@HauteSpot] routing ospf network> print
Flags: X - disabled
  #    NETWORK            AREA
  0    10.10.1.0/24       backbone
[admin@HauteSpot] routing ospf>
```

Interfaces

Submenu level: ***/routing ospf interface***

Description

This facility provides tools for additional in-depth configuration of OSPF interface specific parameters. You do not have to configure interfaces in order to run OSPF

Property Description

authentication-key (*text*; default: **""**) - authentication key have to be used by neighboring routers that are using OSPF's simple password authentication
cost (*integer*: 1..65535; default: **1**) - interface cost expressed as link state metric
dead-interval (*time*; default: **40s**) - specifies the interval after which a neighbor is declared as dead. The interval is advertised in the router's hello packets. This value must be the same for all routers and access servers on a specific network
hello-interval (*time*; default: **10s**) - the interval between hello packets that the router sends on the interface. The smaller the hello-interval, the faster topological changes will be detected, but more routing traffic will ensue. This value must be the same on each end of the adjancency otherwise the adjacency will not form
interface (*name*; default: **all**) - interface on which OSPF will run
all - is used for the interfaces not having any specific settings
priority (*integer*: 0..255; default: **1**) - router's priority. It helps to determine the designated router for the network. When two routers attached to a network both attempt to become the designated router, the one with the higher router's priority takes precedence
retransmit-interval (*time*; default: **5s**) - time between retransmitting lost link state advertisements. When a router sends a link state advertisement (LSA) to its neighbor, it keeps the LSA until it receives back the acknowledgment. If it receives no acknowledgment in time, it will retransmit the LSA. The following settings are recommended: for Broadcast network are 5 seconds and for Point-to-Point network are 10 seconds
transmit-delay (*time*; default: **1s**) - link state transmit delay is the estimated time it takes to transmit a link state update packet on the interface

Example

To add an entry that specifies that **ether2** interface should send Hello packets every 5 seconds, do the following:

```
[admin@HauteSpot] routing ospf> interface add interface=ether2 hello-
interval=5s
[admin@HauteSpot] routing ospf> interface print
  0 interface=ether2 cost=1 priority=1 authentication-key=""
    retransmit-interval=5s transmit-delay=1s hello-interval=5s
    dead-interval=40s


[admin@HauteSpot] routing ospf>
```

Virtual Links

Submenu level: ***/routing ospf virtual-link***

Description

As stated in OSPF RFC, the backbone area must be contiguous. However, it is possible to define areas in such a way that the backbone is no longer contiguous. In this case the system administrator must restore backbone connectivity by configuring virtual links. Virtual link can be configured between two routers through common area called transit area, one of them should have to be connected with backbone. Virtual links belong to the backbone. The protocol treats two routers joined by a virtual link as if they were connected by an unnumbered point-to-point network

Property Description

neighbor-id (*IP address*; default: **0.0.0.0**) - specifies **router-id** of the neighbour
transit-area (*name*; default: **(unknown)**) - a non-backbone area the two routers have in common

Notes

Virtual links can not be estabilished through stub areas

Example

To add a virtual link with the 10.0.0.201 router through the ex area, do the following:

```
[admin@HauteSpot] routing ospf virtual-link> add neighbor-id=10.0.0.201 \
\... transit-area=ex
[admin@HauteSpot] routing ospf virtual-link> print
Flags: X - disabled, I - invalid
  #   NEIGHBOR-ID     TRANSIT-AREA
  0   10.0.0.201      ex
[admin@HauteSpot] routing ospf virtual-link>
```

Virtual link should be configured on both routers

Neighbours

Submenu level: ***/routing ospf neigbor***

Description

The submenu provides an access to the list of OSPF neighbors, *id est* the routers adjacent to the current router, and supplies brief statistics

Property Description

address (*read-only: IP address*) - appropriate IP address of the neighbour
backup-dr-id (*read-only: IP address*) - backup designated router's router id for this neighbor
db-summaries (*read-only: integer*) - number of records in link-state database advertised by the neighbour
dr-id (*read-only: IP address*) - designated router's router id for this neighbor
ls-requests (*read-only: integer*) - number of link-state requests
ls-retransmits (*read-only: integer*) - number of link-state retransmits
priority (*read-only: integer*) - the priority of the neigbour which is used in designated router elections via Hello protocol on this network
router-id (*read-only: IP address*) - the **router-id** parameter of the neighbour
state (*read-only:* Down | Attempt | Init | 2-Way | ExStart | Exchange | Loading | Full) - the state of the connection:
Down - the connection is down
Attempt - the router is sending Hello protocol packets
Init - Hello packets are exchanged between routers to create a Neighbour Relationship
2-Way - the routers add each other to their Neighbour database and they become neighbours
ExStart - the DR (Designated Router) and BDR (Backup Designated Router) create an adjancency with each other and they begin creating their link-state databases using Database Description Packets
Exchange - is the process of discovering routes by exchanging Database Description Packets
Loading - receiving information from the neighbour
Full - the link-state databases are completely synchronized. The routers are routing traffic and continue sending each other hello packets to maintain the adjacency and the routing information
state-changes (*read-only: integer*) - number of connection state changes

Notes

The neighbour's list also displays the router itself with 2-Way state

Example

The following text can be observed just after adding an OSPF network:

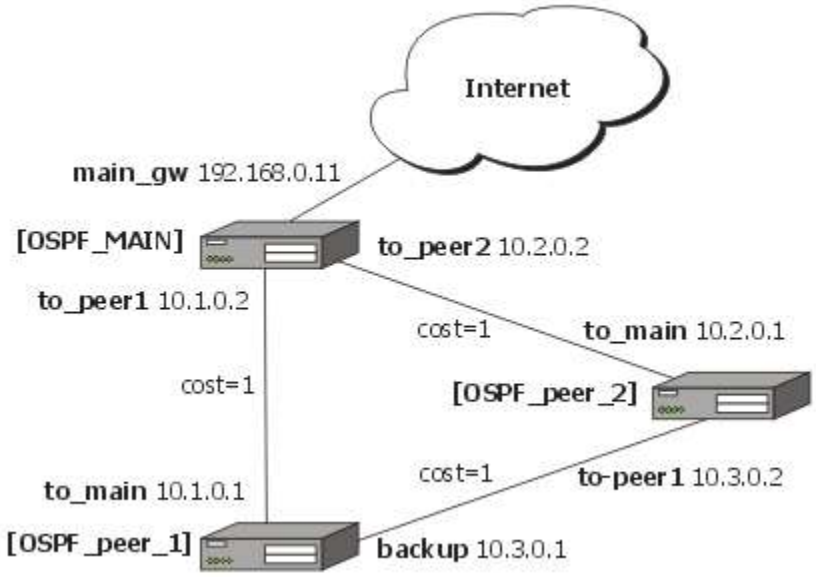
```
admin@HauteSpot] routing ospf> neighbor print
 router-id=10.0.0.204 address=10.0.0.204 priority=1 state="2-Way"
    state-changes=0 ls-retransmits=0 ls-requests=0 db-summaries=0
    dr-id=0.0.0.0 backup-dr-id=0.0.0.0

[admin@HauteSpot] routing ospf>
```

Application Examples

OSPF backup without using a tunnel

Let us assume that the link between the routers OSPF-Main and OSPF-peer-1 is the main one. If it goes down, we want the traffic switch over to the link going through the router OSPF-peer-2.

This example shows how to use OSPF for backup purposes, if you are controlling all the involved routers, and you can run OSPF on them



For this:

1. We introduce an OSPF area with area ID=0.0.0.1, which includes all three routers shown on the diagram
2. Only the OSPF-Main router will have the default route configured. Its interfaces peer1 and peer2 will be configured for the OSPF protocol. The interface main_gw will not be used for distributing the OSPF routing information
3. The routers OSPF-peer-1 and OSPF-peer-2 will distribute their connected route information, and receive the default route using the OSPF protocol

Now let's setup the **OSPF_MAIN** router.

The router should have 3 NICs:

```
[admin@OSPF_MAIN] interface> print
Flags: X - disabled, D - dynamic, R - running
  #      NAME                                    TYPE                    RX-
RATE      TX-RATE      MTU
  0   R main_gw                                  ether                     0
0          1500
  1   R to_peer_1                                ether                     0
0          1500
```

```
  2   R to_peer_2                                ether                    0
0          1500
```

Add all needed ip addresses to interfaces as it is shown here:

```
[admin@OSPF_MAIN] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
  #    ADDRESS                 NETWORK             BROADCAST
INTERFACE
  0    192.168.0.11/24         192.168.0.0         192.168.0.255      main_gw
  1    10.1.0.2/24             10.1.0.0            10.1.0.255         to_peer_1
  2    10.2.0.2/24             10.2.0.0            10.2.0.255         to_peer_2
```

You should set distribute-default as if-installed-as-type-2, redistribute-connected as as-type-1 and redistribute-static as as-type-2. Metric-connected, metric-static, metric-rip, metric-bgp should be zero

```
[admin@OSPF_MAIN] routing ospf> print
                          router-id: 0.0.0.0
                 distribute-default: if-installed-as-type-2
              redistribute-connected: as-type-1
                 redistribute-static: as-type-2
                    redistribute-rip: no
                    redistribute-bgp: no
                      metric-default: 1
                    metric-connected: 0
                       metric-static: 0
                          metric-rip: 0
                          metric-bgp: 0
```

Define new OSPF area named local_10 with area-id 0.0.0.1:

```
[admin@OSPF_MAIN] routing ospf area> print
Flags: X - disabled, I - invalid
  #    NAME                                AREA-ID             STUB DEFAULT-
COST AUTHENTICATION
  0    backbone                            0.0.0.0
none
  1    local_10                            0.0.0.1              no         1
none
```

Add connected networks with area local_10 in ospf network:

```
[admin@OSPF_MAIN] routing ospf network> print
Flags: X - disabled, I - invalid
  #    NETWORK                 AREA
  0    10.1.0.0/24             local_10
  1    10.2.0.0/24             local_10
```

For main router the configuration is done. Next, you should configure **OSPF_peer_1** router

Enable followong interfaces on **OSPF_peer_1**:

```
[admin@OSPF_peer_1] interface> print
Flags: X - disabled, D - dynamic, R - running
  #      NAME                                    TYPE                    RX-
RATE      TX-RATE      MTU
  0   R backup                                  ether                     0
0         1500
  1   R to_main                                 ether                     0
0         1500
```

Assign IP addresses to these interfaces:

```
[admin@OSPF_peer_1] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
  #    ADDRESS                 NETWORK             BROADCAST
INTERFACE
  0    10.1.0.1/24             10.1.0.0            10.1.0.255          to_main
  1    10.3.0.1/24             10.3.0.0            10.3.0.255          backup
```

Set redistribute-connected as as-type-1. Metric-connected, metric-static, metric-rip, metric-bgp should be zero.

```
[admin@OSPF_peer_1] routing ospf> print
                          router-id: 0.0.0.0
                 distribute-default: never
              redistribute-connected: as-type-1
                 redistribute-static: no
                    redistribute-rip: no
                    redistribute-bgp: no
                      metric-default: 1
                    metric-connected: 0
                       metric-static: 0
                          metric-rip: 0
                          metric-bgp: 0
```

Add the same area as in main router:

```
[admin@OSPF_peer_1] routing ospf area> print
Flags: X - disabled, I - invalid
  #    NAME                                AREA-ID             STUB DEFAULT-
COST AUTHENTICATION
  0    backbone                            0.0.0.0
none
  1    local_10                            0.0.0.1              no         1
none
```

Add connected networks with area local_10:

```
[admin@OSPF_peer_1] routing ospf network> print
Flags: X - disabled, I - invalid
  #    NETWORK                 AREA
  0    10.3.0.0/24             local_10
  1    10.1.0.0/24             local_10
```

Finally, set up the **OSPF_peer_2** router. Enable the following interfaces:

```
[admin@OSPF_peer_2] interface> print
Flags: X - disabled, D - dynamic, R - running
  #      NAME                                TYPE                    RX-
RATE      TX-RATE      MTU
  0   R to_main                              ether                     0
0         1500
  1   R to_peer_1                            ether                     0
0         1500
```

Add the needed IP addresses:

```
[admin@OSPF_peer_2] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
  #    ADDRESS                 NETWORK             BROADCAST
INTERFACE
  0    10.2.0.1/24             10.2.0.0            10.2.0.255         to_main
  1    10.3.0.2/24             10.3.0.0            10.3.0.255
to_peer_1
```

Add the same area as in previous routers:

```
[admin@OSPF_peer_2] routing ospf area> print
Flags: X - disabled, I - invalid
  #    NAME                                AREA-ID             STUB DEFAULT-
COST AUTHENTICATION
  0    backbone                            0.0.0.0
none
  1    local_10                            0.0.0.1              no         1
none
```

Add connected networks with the same area:

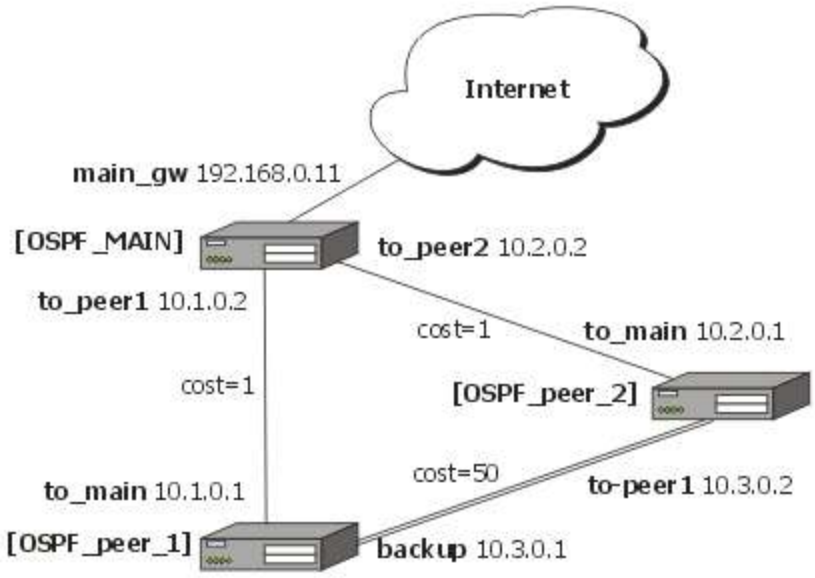
```
[admin@OSPF_peer_2] routing ospf network> print
Flags: X - disabled, I - invalid
  #    NETWORK                 AREA
  0    10.2.0.0/24             local_10
  1    10.3.0.0/24             local_10
```

After all routers have been set up as described above, and the links between them are operational, the routing tables of the three routers look as follows:

```
[admin@OSPF_MAIN] ip route> print
Flags: X - disabled, I - invalid, D - dynamic, J - rejected,
C - connect, S - static, r - rip, o - ospf, b - bgp
  #      DST-ADDRESS            G GATEWAY              DISTANCE INTERFACE
  0 Io 192.168.0.0/24                                     110
  1 DC 192.168.0.0/24         r 0.0.0.0                  0        main_gw
  2 Do 10.3.0.0/24            r 10.2.0.1                 110      to_peer_2
                              r 10.1.0.1                          to_peer_1
  3 Io 10.2.0.0/24                                        110
  4 DC 10.2.0.0/24            r 0.0.0.0                  0        to_peer_2
  5 Io 10.1.0.0/24                                        110
  6 DC 10.1.0.0/24            r 0.0.0.0                  0        to_peer_1
[admin@OSPF_peer_1] ip route> print
Flags: X - disabled, I - invalid, D - dynamic, J - rejected,
C - connect, S - static, r - rip, o - ospf, b - bgp
  #      DST-ADDRESS            G GATEWAY              DISTANCE INTERFACE
  0 Do 192.168.0.0/24         r 10.1.0.2                 110      to_main
  1 Io 10.3.0.0/24                                        110
  2 DC 10.3.0.0/24            r 0.0.0.0                  0        backup
  3 Do 10.2.0.0/24            r 10.1.0.2                 110      to_main
                              r 10.3.0.2                          backup
  4 Io 10.1.0.0/24                                        110
  5 DC 10.1.0.0/24            r 0.0.0.0                  0        to_main
[admin@OSPF_peer_2] ip route> print
Flags: X - disabled, I - invalid, D - dynamic, J - rejected,
C - connect, S - static, r - rip, o - ospf, b - bgp
  #      DST-ADDRESS            G GATEWAY              DISTANCE INTERFACE
  0 Do 192.168.0.0/24         r 10.2.0.2                 110      to_main
  1 Io 10.3.0.0/24                                        110
  2 DC 10.3.0.0/24            r 0.0.0.0                  0        to_peer_1
  3 Io 10.2.0.0/24                                        110
  4 DC 10.2.0.0/24            r 0.0.0.0                  0        to_main
  5 Do 10.1.0.0/24            r 10.3.0.1                 110      to_peer_1
                              r 10.2.0.2                          to_main
```

Routing tables with Revised Link Cost

This example shows how to set up link cost. Let us assume, that the link between the routers **OSPF_peer_1** and **OSPF_peer_2** has a higher cost (might be slower, we have to pay more for the traffic through it, etc.).



We should change cost value in both routers: **OSPF_peer_1** and **OSPF_peer_2** to 50. To do this, we need to add a following interface:

```
[admin@OSPF_peer_1] routing ospf interface> add interface=backup cost=50
[admin@OSPF_peer_1] routing ospf interface> print
  0 interface=backup cost=50 priority=1 authentication-key=""
 retransmit-interval=5s transmit-delay=1s hello-interval=10s
dead-interval=40s

[admin@OSPF_peer_2] routing ospf interface> add interface=to_peer_1 cost=50
[admin@OSPF_peer_2] routing ospf interface> print
  0 interface=to_peer_1 cost=50 priority=1 authentication-key=""
 retransmit-interval=5s transmit-delay=1s hello-interval=10s
     dead-interval=40s
```

After changing the cost settings, we have only one equal cost multipath route left - to the network 10.3.0.0/24 from **OSPF_MAIN** router.

Routes on **OSPF_MAIN** router:

```
[admin@OSPF_MAIN] ip route> print
Flags: X - disabled, I - invalid, D - dynamic, J - rejected,
C - connect, S - static, r - rip, o - ospf, b - bgp
  #      DST-ADDRESS            G GATEWAY             DISTANCE INTERFACE
  0 Io 192.168.0.0/24                                     110
  1 DC 192.168.0.0/24         r 0.0.0.0                   0       main_gw
  2 Do 10.3.0.0/24            r 10.2.0.1                  110     to_peer_2
                              r 10.1.0.1                          to_peer_1
  3 Io 10.2.0.0/24                                        110
  4 DC 10.2.0.0/24            r 0.0.0.0                   0       to_peer_2
  5 Io 10.1.0.0/24                                        110
```

```
  6 DC 10.1.0.0/24            r 0.0.0.0                0       to_peer_1
```

On **OSPF_peer_1**:

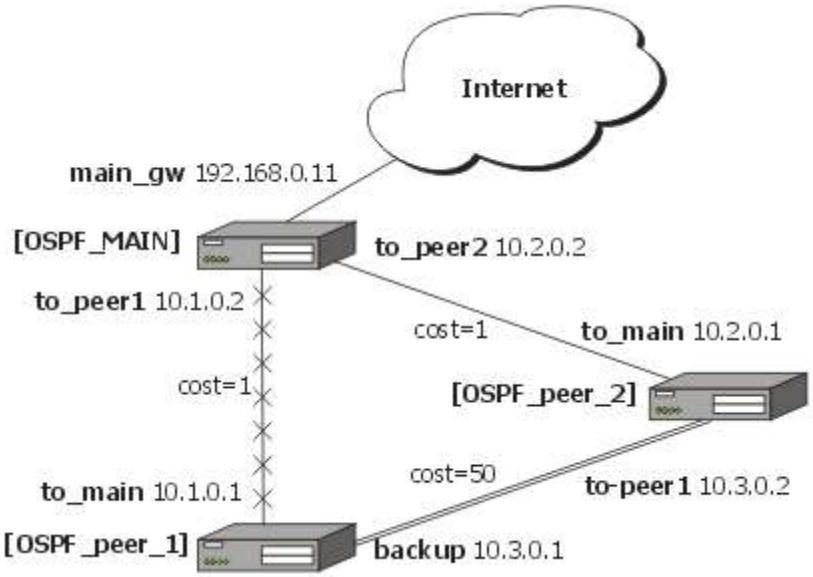
```
[admin@OSPF_peer_1] > ip route pr
Flags: X - disabled, I - invalid, D - dynamic, J - rejected,
C - connect, S - static, r - rip, o - ospf, b - bgp
  #      DST-ADDRESS            G GATEWAY             DISTANCE INTERFACE
  0 Do 192.168.0.0/24         r 10.1.0.2               110       to_main
  1 Io 10.3.0.0/24                                      110
  2 DC 10.3.0.0/24            r 0.0.0.0                0         backup

  3 Do 10.2.0.0/24            r 10.1.0.2               110       to_main
  4 Io 10.1.0.0/24                                      110
  5 DC 10.1.0.0/24            r 0.0.0.0                0         to_main
```

On **OSPF_peer_2**:

```
[admin@OSPF_peer_2] > ip route print
Flags: X - disabled, I - invalid, D - dynamic, J - rejected,
C - connect, S - static, r - rip, o - ospf, b - bgp
  #      DST-ADDRESS            G GATEWAY             DISTANCE INTERFACE
  0 Do 192.168.0.0/24         r 10.2.0.2               110       to_main
  1 Io 10.3.0.0/24                                      110
  2 DC 10.3.0.0/24            r 0.0.0.0                0         to_peer_1
  3 Io 10.2.0.0/24                                      110
  4 DC 10.2.0.0/24            r 0.0.0.0                0         to_main
  5 Do 10.1.0.0/24            r 10.2.0.2               110       to_main
```

Functioning of the Backup

If the link between routers **OSPF_MAIN** and **OSPF_peer_1** goes down, we have the following situation:



The OSPF routing changes as follows:

Routes on **OSPF_MAIN** router:

```
[admin@OSPF_MAIN] ip route> print
Flags: X - disabled, I - invalid, D - dynamic, J - rejected,
C - connect, S - static, r - rip, o - ospf, b - bgp
  #      DST-ADDRESS            G GATEWAY             DISTANCE INTERFACE
  0 Io 192.168.0.0/24                                     110
  1 DC 192.168.0.0/24         r 0.0.0.0                  0         main_gw
  2 Do 10.3.0.0/24            r 10.2.0.1                 110       to_peer_2
  3 Io 10.2.0.0/24                                        110
  4 DC 10.2.0.0/24            r 0.0.0.0                  0         to_peer_2
  5 Io 10.1.0.0/24                                        110
  6 DC 10.1.0.0/24            r 0.0.0.0                  0         to_peer_1
```

On **OSPF_peer_1**:

```
[admin@OSPF_peer_1] ip route> print
Flags: X - disabled, I - invalid, D - dynamic, J - rejected,
C - connect, S - static, r - rip, o - ospf, b - bgp
  #      DST-ADDRESS            G GATEWAY             DISTANCE INTERFACE
  0 Do 192.168.0.0/24         r 10.3.0.2                 110       backup
  1 Io 192.168.0.0/24                                     110
  2 DC 10.3.0.0/24            r 0.0.0.0                  0         backup
  3 Do 10.2.0.0/24            r 10.3.0.2                 110       backup
  4 Io 10.1.0.0/24                                        110
  5 DC 10.1.0.0/24            r 0.0.0.0                  0         to_main
```

On **OSPF_peer_2**:

```
[admin@OSPF_peer_2] ip route> print
Flags: X - disabled, I - invalid, D - dynamic, J - rejected,
C - connect, S - static, r - rip, o - ospf, b - bgp
  #      DST-ADDRESS           G GATEWAY             DISTANCE INTERFACE
  0 Do 192.168.0.0/24         r 10.2.0.2              110      to_main
  1 Io 10.3.0.0/24                                     110
  2 DC 10.3.0.0/24            r 0.0.0.0               0        to_peer_1
  3 Io 10.2.0.0/24                                     110
  4 DC 10.2.0.0/24            r 0.0.0.0               0        to_main
  5 Do 10.1.0.0/24            r 10.2.0.2              110      to_main
```

The change of the routing takes approximately 40 seconds (the hello-interval setting). If required, this setting can be adjusted, but it should be done on all routers within the OSPF area!

HauteRouterOS Packet Filtering

Document revision: 2.7 (Tuesday, October 30, 2007)

Applies to: HauteRouterOS V3.0

General Information

Summary

The firewall implements packet filtering and thereby provides security functions that are used to manage data flow to, from and through the router. Along with the Network Address Translation it serve as a tool for preventing unauthorized access to directly attached networks and the router itself as well as a filter for outgoing traffic.

Quick Setup Guide

- To add a firewall rule which drops all **TCP** packets that are destined to port **135** and going through the router, use the following command:

```
/ip firewall filter add chain=forward dst-port=135 protocol=tcp
action=drop
```

- To deny acces to the router via Telnet (protocol TCP, port 23), type the following command:

```
/ip firewall filter add chain=input protocol=tcp dst-port=23
action=drop
```

- To only allow not more than 5 simultaneous connections from each of the clients, do the following:

```
/ip firewall filter add chain=forward protocol=tcp tcp-flags=syn
connection-limit=6,32 action=drop
```

Specifications

Packages required: ***system***
License required: *Level1 (P2P filters limited to 1) , Level3*
Submenu level: ***/ip firewall filter***
Standards and Technologies: IP, RFC2113
Hardware usage: *Increases with filtering rules count*

Firewall Filter

Submenu level: ***/ip firewall filter***

Description

Network firewalls keep outside threats away from sensitive data available inside the network. Whenever different networks are joined together, there is always a threat that someone from outside of your network will break into your LAN. Such break-ins may result in private data being stolen and distributed, valuable data being altered or destroyed, or entire hard drives being erased. Firewalls are used as a means of preventing or minimizing the security risks inherent in connecting to other networks. Properly configured firewall plays a key role in efficient and secure network infrastrure deployment.

HauteRouterOS has very powerful firewall implementation with features including:

- stateful packet filtering
- peer-to-peer protocols filtering
- traffic classification by:
  - source MAC address
  - IP addresses (network or list) and address types (broadcast, local, multicast, unicast)
  - port or port range
  - IP protocols
  - protocol options (ICMP type and code fields, TCP flags, IP options and MSS)
  - interface the packet arrived from or left through

- internal flow and connection marks
- ToS (DSCP) byte
- packet content
- rate at which packets arrive and sequence numbers
- packet size
- packet arrival time
- and much more!

General Filtering Principles

The firewall operates by means of firewall rules. A rule is a definitive form expression that tells the router what to do with a particular IP packet. Each rule consists of two parts that are the matcher which matches traffic flow against given conditions and the action which defines what to do with the mathched packets. Rules are organized in chains for better management.

The filter facility has three default chains: **input**, **forward** and **output** that are responsible for traffic coming from, throurh and to the router, respectively. New user-defined chains can be added, as necessary. Since these chains have no default traffic to match, rules with **action=jump** and relevant **jump-target** should be added to one or more of the three default chains.

Filter Chains

As mentioned before, the firewall filtering rules are grouped together in chains. It allows a packet to be matched against one common criterion in one chain, and then passed over for processing against some other common criteria to another chain. For example a packet should be matched against the **IP address:port** pair. Of course, it could be achieved by adding as many rules with **IP address:port** match as required to the **forward** chain, but a better way could be to add one rule that matches traffic from a particular IP address, e.g.: `/ip firewall filter add src-address=1.1.1.2/32 jump-target="mychain"` and in case of successfull match passes control over the IP packet to some other chain, *id est* **mychain** in this example. Then rules that perform matching against separate ports can be added to **mychain** chain without specifying the IP addresses.

There are three predefined chains, which cannot be deleted:

- **input** - used to process packets entering the router through one of the interfaces with the destination IP address which is one of the router's addresses. Packets passing through the router are not processed against the rules of the **input** chain
- **forward** - used to process packets passing through the router
- **output** - used to process packets originated from the router and leaving it through one of the interfaces. Packets passing through the router are not processed against the rules of the **output** chain

When processing a chain, rules are taken from the chain in the order they are listed there from top to bottom. If a packet matches the criteria of the rule, then the

specified action is performed on it, and no more rules are processed in that chain (the exception is the **passthrough** action). If a packet has not matched any rule within the chain, then it is accepted.

Property Description

action (accept | add-dst-to-address-list | add-src-to-address-list | drop | jump | log | passthrough | reject | return | tarpit; default: **accept**) - action to undertake if the packet matches the rule
accept - accept the packet. No action is taken, i.e. the packet is passed through and no more rules are applied to it
add-dst-to-address-list - adds destination address of an IP packet to the address list specified by **address-list** parameter
add-src-to-address-list - adds source address of an IP packet to the address list specified by **address-list** parameter
drop - silently drop the packet (without sending the ICMP reject message)
jump - jump to the chain specified by the value of the **jump-target** parameter
log - each match with this action will add a message to the system log
passthrough - ignores this rule and goes on to the next one
reject - reject the packet and send an ICMP reject message
return - passes control back to the chain from where the jump took place
tarpit - captures and holds incoming TCP connections (replies with SYN/ACK to the inbound TCP SYN packet)
address-list (*name*) - specifies the name of the address list to collect IP addresses from rules having **action=add-dst-to-address-list** or **action=add-src-to-address-list** actions. These address lists could be later used for packet matching
address-list-timeout (*time*; default: **00:00:00**) - time interval after which the address will be removed from the address list specified by **address-list** parameter. Used in conjunction with **add-dst-to-address-list** or **add-src-to-address-list** actions
00:00:00 - leave the address in the address list forever
chain (forward | input | output | *name*) - specifies the chain to put a particular rule into. As the different traffic is passed through different chains, always be careful in choosing the right chain for a new rule. If the input does not match the name of an already defined chain, a new chain will be created
comment (*text*) - a descriptive comment for the rule. A comment can be used to identify rules form scripts
connection-bytes (*integer-integer*) - matches packets only if a given amount of bytes has been transfered through the particular connection
0 - means infinity, *exempli gratia*: **connection-bytes=2000000-0** means that the rule matches if more than 2MB has been transfered through the relevant connection
connection-limit (*integer*,*netmask*) - restrict connection limit per address or address block
connection-mark (*name*) - matches packets marked via mangle facility with particular connection mark
connection-state (estabilished | invalid | new | related) - interprets the connection tracking analysis data for a particular packet
estabilished - a packet which belongs to an existing connection, *exempli gratia* a reply packet or a packet which belongs to already replied connection
invalid - a packet which could not be identified for some reason. This includes out of memory condition and ICMP errors which do not correspond to any known connection. It is generally advised to drop these packets
new - a packet which begins a new TCP connection
related - a packet which is related to, but not part of an existing connection, such as ICMP errors

or a packet which begins FTP data connection (the later requires enabled FTP connection tracking helper under **/ip firewall service-port**)

connection-type (ftp | gre | h323 | irc | mms | pptp | quake3 | tftp) - matches packets from related connections based on information from their connection tracking helpers. A relevant connection helper must be enabled under **/ip firewall service-port**

content (*text*) - the text packets should contain in order to match the rule

dst-address (*IP address*/*netmask* | *IP address-IP address*) - specifies the address range an IP packet is destined to. Note that console converts entered **address/netmask** value to a valid network address, i.e.:**1.1.1.1/24** is converted to **1.1.1.0/24**

dst-address-list (*name*) - matches destination address of a packet against user-defined address list

dst-address-type (unicast | local | broadcast | multicast) - matches destination address type of the IP packet, one of the:

unicast - IP addresses used for one point to another point transmission. There is only one sender and one receiver in this case

local - matches addresses assigned to router's interfaces

broadcast - the IP packet is sent from one point to all other points in the IP subnetwork

multicast - this type of IP addressing is responsible for transmission from one or more points to a set of other points

dst-limit (*integer*/*time*{0,1},*integer*,dst-address | dst-port | src-address{+},*time*{0,1}) - limits the packet per second (pps) rate on a per destination IP or per destination port base. As opposed to the **limit** match, every destination IP address / destination port has it's own limit. The options are as follows (in order of appearance):

Count - maximum average packet rate, measured in packets per second (pps), unless followed by **Time** option

Time - specifies the time interval over which the packet rate is measured

Burst - number of packets to match in a burst

Mode - the classifier(-s) for packet rate limiting

Expire - specifies interval after which recorded IP addresses / ports will be deleted

dst-port (*integer*: 0..65535-*integer*: 0..65535{*}) - destination port number or range

hotspot (*multiple choice:* from-client | auth | local-dst | http) - matches packets received from clients against various Hot-Spot. All values can be negated

from-client - true, if a packet comes from HotSpot client

auth - true, if a packet comes from authenticted client

local-dst - true, if a packet has local destination IP address

hotspot - true, if it is a TCP packet from client and either the transparent proxy on port 80 is enabled or the client has a proxy address configured and this address is equal to the address:port pair of the IP packet

icmp-options (*integer*:*integer*) - matches ICMP Type:Code fields

in-interface (*name*) - interface the packet has entered the router through

ipv4-options (any | loose-source-routing | no-record-route | no-router-alert | no-source-routing | no-timestamp | none | record-route | router-alert | strict-source-routing | timestamp) - match ipv4 header options

any - match packet with at least one of the ipv4 options

loose-source-routing - match packets with loose source routing option. This option is used to route the internet datagram based on information supplied by the source

no-record-route - match packets with no record route option. This option is used to route the internet datagram based on information supplied by the source

no-router-alert - match packets with no router alter option

no-source-routing - match packets with no source routing option

no-timestamp - match packets with no timestamp option

record-route - match packets with record route option
router-alert - match packets with router alter option
strict-source-routing - match packets with strict source routing option
timestamp - match packets with timestamp
jump-target (forward | input | output | *name*) - name of the target chain to jump to, if the **action=jump** is used
limit (*integer*/*time*{0,1},*integer*) - restricts packet match rate to a given limit. Usefull to reduce the amount of log messages
Count - maximum average packet rate, measured in packets per second (pps), unless followed by **Time** option
Time - specifies the time interval over which the packet rate is measured
Burst - number of packets to match in a burst
log-prefix (*text*) - all messages written to logs will contain the prefix specified herein. Used in conjunction with **action=log**
nth (*integer*,*integer*: 0..15,*integer*{0,1}) - match a particular Nth packet received by the rule. One of 16 available counters can be used to count packets
Every - match every **Every+1**th packet. For example, if **Every=1** then the rule matches every 2nd packet
Counter - specifies which counter to use. A counter increments each time the rule containing **nth** match matches
Packet - match on the given packet number. The value by obvious reasons must be between **0** and **Every**. If this option is used for a given counter, then there must be at least **Every+1** rules with this option, covering all values between **0** and **Every** inclusively.
out-interface (*name*) - interface the packet will leave the router through
p2p (all-p2p | bit-torrent | blubster | direct-connect | edonkey | fasttrack | gnutella | soulseek | warez | winmx) - matches packets from various peer-to-peer (P2P) protocols
packet-mark (*text*) - matches packets marked via mangle facility with particular packet mark
packet-size (*integer*: 0..65535-*integer*: 0..65535{0,1}) - matches packet of the specified size or size range in bytes
Min - specifies lower boundary of the size range or a standalone value
Max - specifies upper boundary of the size range
phys-in-interface (*name*) - matches the bridge port physical input device added to a bridge device. It is only useful if the packet has arrived through the bridge
phys-out-interface (*name*) - matches the bridge port physical output device added to a bridge device. It is only useful if the packet will leave the router through the bridge
protocol (ddp | egp | encap | ggp | gre | hmp | icmp | idrp-cmtp | igmp | ipencap | ipip | ipsec-ah | ipsec-esp | iso-tp4 | ospf | pup | rdp | rspf | st | tcp | udp | vmtp | xns-idp | xtp | *integer*) - matches particular IP protocol specified by protocol name or number. You should specify this setting if you want to specify ports
psd (*integer*,*time*,*integer*,*integer*) - attempts to detect TCP and UDP scans. It is advised to assign lower weight to ports with high numbers to reduce the frequency of false positives, such as from passive mode FTP transfers
WeightThreshold - total weight of the latest TCP/UDP packets with different destination ports coming from the same host to be treated as port scan sequence
DelayThreshold - delay for the packets with different destination ports coming from the same host to be treated as possible port scan subsequence
LowPortWeight - weight of the packets with privileged (<=1024) destination port
HighPortWeight - weight of the packet with non-priviliged destination port
random (*integer*: 1..99) - matches packets randomly with given propability
reject-with (icmp-admin-prohibited | icmp-echo-reply | icmp-host-prohibited | icmp-host-unreachable | icmp-net-prohibited | icmp-network-unreachable | icmp-

port-unreachable | icmp-protocol-unreachable | tcp-reset | *integer*) - alters the reply packet of **reject** action
routing-mark (*name*) - matches packets marked by mangle facility with particular routing mark
src-address (*IP address/netmask* | *IP address-IP address*) - specifies the address range an IP packet is originated from. Note that console converts entered **address/netmask** value to a valid network address, i.e.:**1.1.1.1/24** is converted to **1.1.1.0/24**
src-address-list (*name*) - matches source address of a packet against user-defined address list
src-address-type (unicast | local | broadcast | multicast) - matches source address type of the IP packet, one of the:
unicast - IP addresses used for one point to another point transmission. There is only one sender and one receiver in this case
local - matches addresses assigned to router's interfaces
broadcast - the IP packet is sent from one point to all other points in the IP subnetwork
multicast - this type of IP addressing is responsible for transmission from one or more points to a set of other points
src-mac-address (*MAC address*) - source MAC address
src-port (*integer*: 0..65535-*integer*: 0..65535{*}) - source port number or range
tcp-flags (ack | cwr | ece | fin | psh | rst | syn | urg) - tcp flags to match
ack - acknowledging data
cwr - congestion window reduced
ece - ECN-echo flag (explicit congestion notification)
fin - close connection
psh - push function
rst - drop connection
syn - new connection
urg - urgent data
tcp-mss (*integer*: 0..65535) - matches TCP MSS value of an IP packet
time (*time-time*,sat | fri | thu | wed | tue | mon | sun{+}) - allows to create filter based on the packets' arrival time and date or, for locally generated packets, departure time and date
tos (max-reliability | max-throughput | min-cost | min-delay | normal) - specifies a match for the value of Type of Service (ToS) field of an IP header
max-reliability - maximize reliability (ToS=4)
max-throughput - maximize throughput (ToS=8)
min-cost - minimize monetary cost (ToS=2)
min-delay - minimize delay (ToS=16)
normal - normal service (ToS=0)

Notes

Because the NAT rules are applied first, it is important to hold this in mind when setting up firewall rules, since the original packets might be already modified by the NAT

Filter Applications

Protect your RouterOS router

To protect your router, you should not only change admin's password but also set up packet filtering. All packets with destination to the router are processed against the ip firewall input chain. Note, that the input chain does not affect packets which are being transferred through the router.

```
/ ip firewall filter
add chain=input connection-state=invalid action=drop \
       comment="Drop Invalid connections"
add chain=input connection-state=established action=accept \
       comment="Allow Established connections"
add chain=input protocol=udp action=accept \
       comment="Allow UDP"
add chain=input protocol=icmp action=accept \
       comment="Allow ICMP"
add chain=input src-address=192.168.0.0/24 action=accept \
       comment="Allow access to router from known network"
add chain=input action=drop comment="Drop anything else"
```

Protecting the Customer's Network

To protect the customer's network, we should check all traffic which goes through router and block unwanted. For icmp, tcp, udp traffic we will create chains, where will be droped all unwanted packets:

```
/ip firewall filter
add chain=forward protocol=tcp connection-state=invalid \
       action=drop comment="drop invalid connections"
add chain=forward connection-state=established action=accept \
       comment="allow already established connections"
add chain=forward connection-state=related action=accept \
       comment="allow related connections"
```

Block IP addreses called "bogons":

```
add chain=forward src-address=0.0.0.0/8 action=drop
add chain=forward dst-address=0.0.0.0/8 action=drop
add chain=forward src-address=127.0.0.0/8 action=drop
add chain=forward dst-address=127.0.0.0/8 action=drop
add chain=forward src-address=224.0.0.0/3 action=drop
add chain=forward dst-address=224.0.0.0/3 action=drop
```

Make jumps to new chains:

```
add chain=forward protocol=tcp action=jump jump-target=tcp
add chain=forward protocol=udp action=jump jump-target=udp
add chain=forward protocol=icmp action=jump jump-target=icmp
```

```

```

Create tcp chain and deny some tcp ports in it:

```
add chain=tcp protocol=tcp dst-port=69 action=drop \
       comment="deny TFTP"
add chain=tcp protocol=tcp dst-port=111 action=drop \
       comment="deny RPC portmapper"
add chain=tcp protocol=tcp dst-port=135 action=drop \
       comment="deny RPC portmapper"
add chain=tcp protocol=tcp dst-port=137-139 action=drop \
       comment="deny NBT"
add chain=tcp protocol=tcp dst-port=445 action=drop \
       comment="deny cifs"
add chain=tcp protocol=tcp dst-port=2049 action=drop comment="deny NFS"
add chain=tcp protocol=tcp dst-port=12345-12346 action=drop comment="deny
NetBus"
add chain=tcp protocol=tcp dst-port=20034 action=drop comment="deny NetBus"
add chain=tcp protocol=tcp dst-port=3133 action=drop comment="deny
BackOriffice"
add chain=tcp protocol=tcp dst-port=67-68 action=drop comment="deny DHCP"
```

Deny udp ports in udp chain:

```
add chain=udp protocol=udp dst-port=69 action=drop comment="deny TFTP"
add chain=udp protocol=udp dst-port=111 action=drop comment="deny PRC
portmapper"
add chain=udp protocol=udp dst-port=135 action=drop comment="deny PRC
portmapper"
add chain=udp protocol=udp dst-port=137-139 action=drop comment="deny NBT"
add chain=udp protocol=udp dst-port=2049 action=drop comment="deny NFS"
add chain=udp protocol=udp dst-port=3133 action=drop comment="deny
BackOriffice"
```

Allow only needed icmp codes in icmp chain:

```
add chain=icmp protocol=icmp icmp-options=0:0 action=accept \
       comment="drop invalid connections"
add chain=icmp protocol=icmp icmp-options=3:0 action=accept \
       comment="allow established connections"
add chain=icmp protocol=icmp icmp-options=3:1 action=accept \
       comment="allow already established connections"
add chain=icmp protocol=icmp icmp-options=4:0 action=accept \
       comment="allow source quench"
add chain=icmp protocol=icmp icmp-options=8:0 action=accept \
       comment="allow echo request"
add chain=icmp protocol=icmp icmp-options=11:0 action=accept \
       comment="allow time exceed"
add chain=icmp protocol=icmp icmp-options=12:0 action=accept \
       comment="allow parameter bad"
```

```
add chain=icmp action=drop comment="deny all other types"
```

Contents

Packet Flow 505
- Diagram 505
- Analysis 506
  - Basic Concepts 506
  - Configurable Facilities 507
  - Automated processes and decisions 507
- Examples 509
  - Bridging with use-ip-firewall=yes 509
  - Routing - from Ethernet to Ethernet interface 509
  - Routing from one Bridge interface to different Bridge interface 510
  - IPsec encryption 511
  - IPsec decryption 511

Queues 511
- Simple Queues 512
  - Flow Identifiers 512
  - Other properties 513
  - HTB Properties 513
  - Statistics 514
- Queue Tree 514
  - Flow Identifiers 515
  - HTB Properties 515
  - Statistics 515
- Queue Types 516
  - PFIFO and BFIFO 516
  - RED 516
  - SFQ 517
  - PCQ 517
- Interface Queue 517

HTB 517
- Theory 517
  - Structure 517
  - Dual Limitation 517
  - Priority 518
- Examples 518
  - Structure 519
  - Example 1 : Usual case 519
  - Result of Example 1 519
  - Example 2 : Usual case with max-limit 520
  - Result of Example 2 520
  - Example 3 : Inner queue limit-at 521
  - Result of Example 3 521
  - Example 4 : Leaf queue limit-at 522
  - Result of Example 4 522

DSCP based QoS with HTB 522
  - DSCP marking/mangling 523
  - Setting up the queue tree 523

Remarks 524

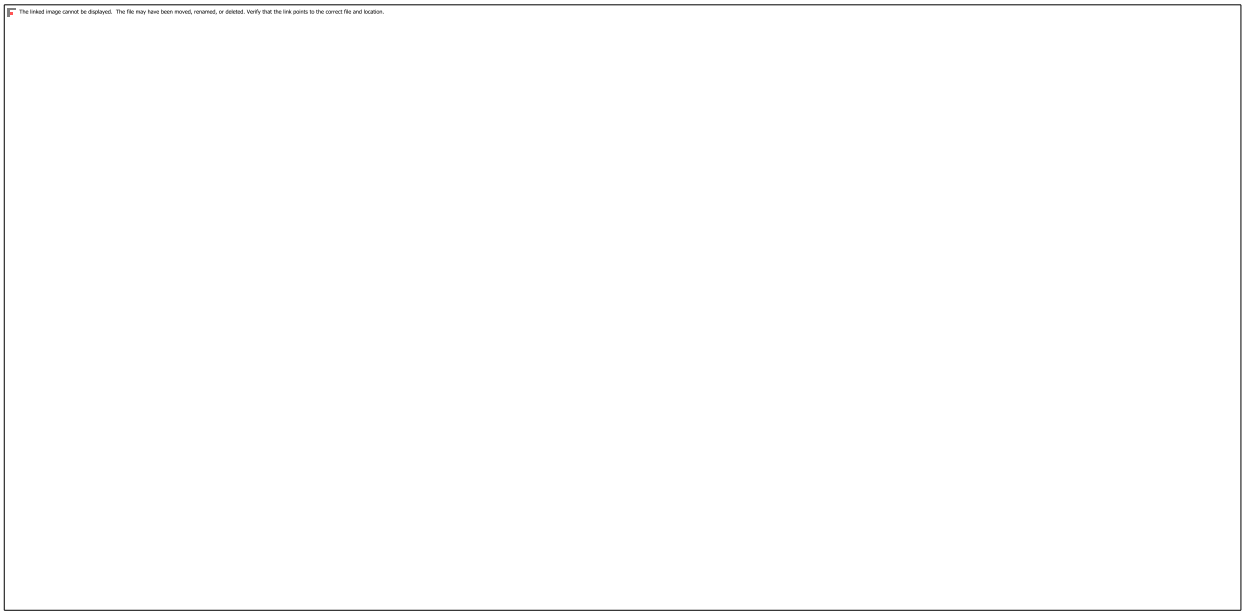
Packet Flow

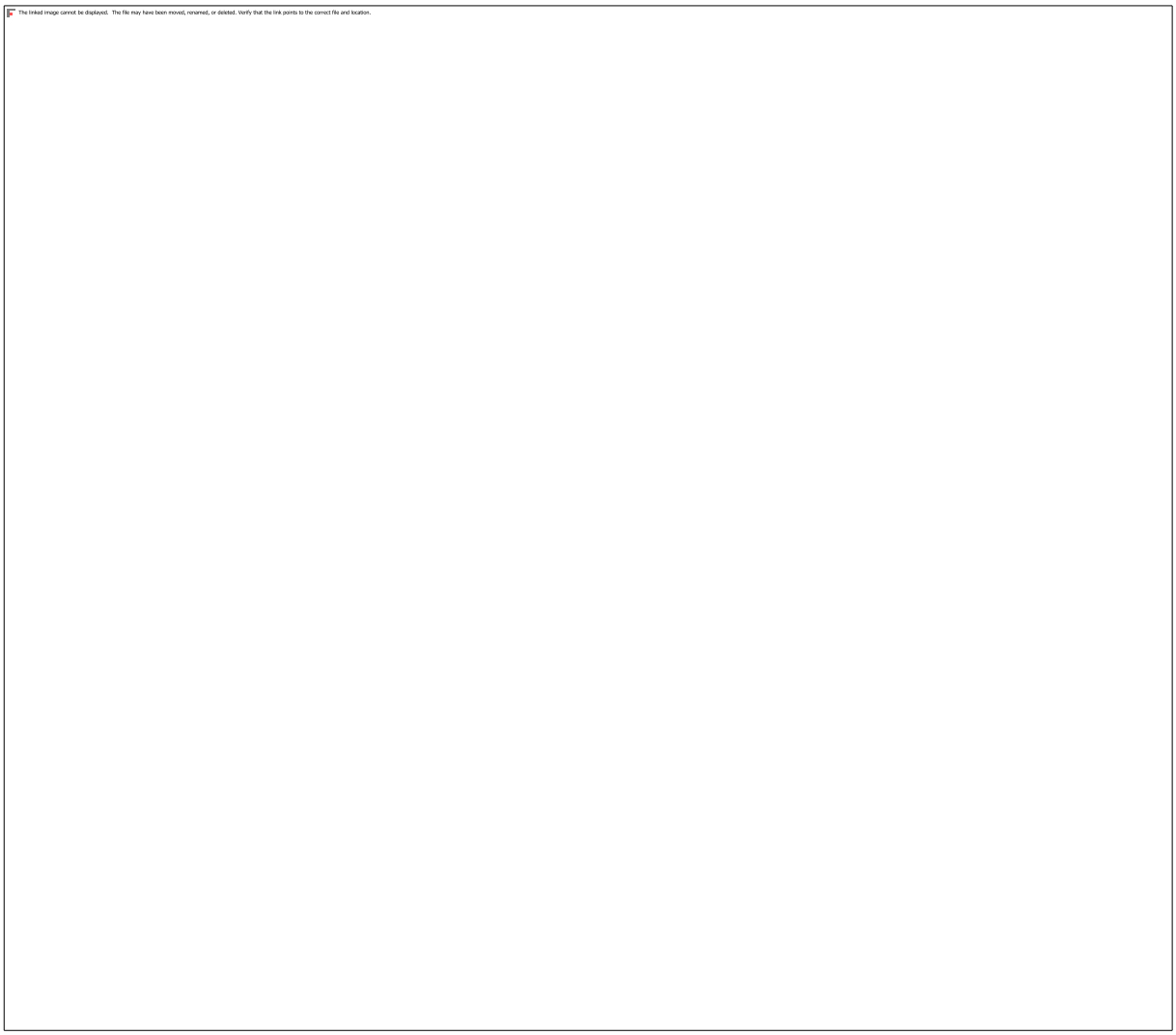
HauteRouterOS is designed to be easy to operate in various aspects of network configuration. Therefore creating limitation for individual IP or natting internal clients to a public address or Hotspot configuration can be done without the knowledge about how the packets are processed in the router - you just go to corresponding menu and create necessary configuration.
However more complicated tasks, such as traffic prioritization, routing policies, where it is necessary to utilize more than one HauteRouterOS facility, requires knowledge: How these facilities work together? What happens when and why?
To address these questions we created a packet flow diagram.

Diagram

As it was impossible to get everything in one diagram, **Packet flow diagram** for HauteRouterOS v3.x was created in 2 parts:

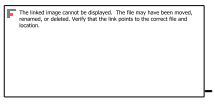
- **Bridging or Layer-2 (MAC)** where Routing part is simplified to one "Layer-3" box
- **Routing or Layer-3 (IP)** where Bridging part is simplified to one "Bridging" box

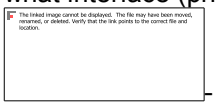

The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

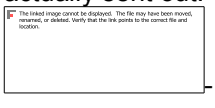


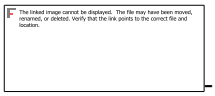
Analysis

Basic Concepts

 - starting point in packets way thought the router facilities. It does not matter what interface (physical or virtual) packet is received it will start its way from here.

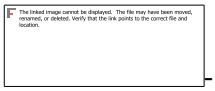
 - last point in packets way thought the router facilities. Just before the packet is actually sent out.

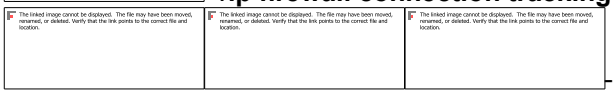
 - last point in packets way **to** router itself, after this packet is discarded

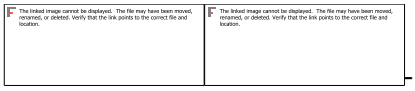
 - starting point for packets generated by router itself

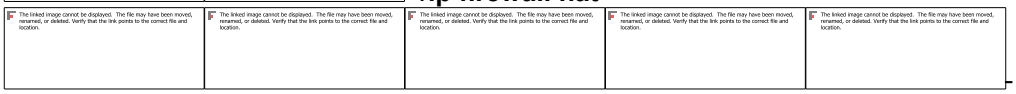
Configurable Facilities

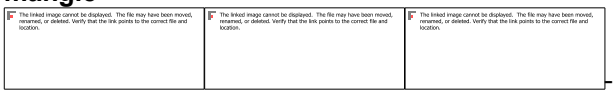
Each and every facilities in this section corresponds with one particular menu in HauteRouterOS. Users are able to access those menu and configure these facilities directly

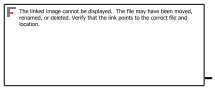
 - **/ip firewall connection tracking**

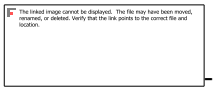
 - **/ip firewall filter**

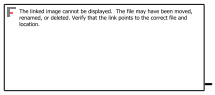
 - **/ip firewall nat**

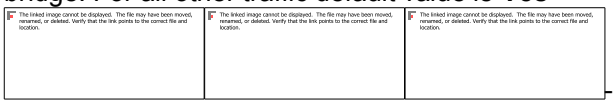
 - **/ip firewall mangle**

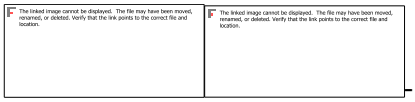
 - **/queue simple** and **/queue tree**

 - **/ip ipsec policy**

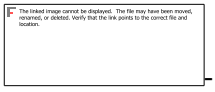
 - **/ip accounting**

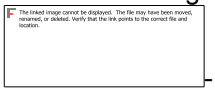
 - **/interface bridge settings** - available only for traffic that go **thought** the bridge. For all other traffic default value is **Yes**

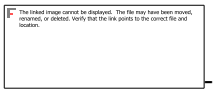
 - **/interface bridge filter**

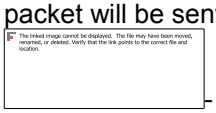
 - **/interface bridge nat**

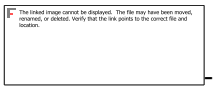
Automated processes and decisions

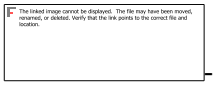
 - check if the *actual input interface* is a port for bridge OR checks if *input interface* is bridge

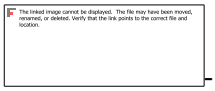
 - allow to capture traffic witch otherwise would be discarded by connection tracking - this way our Hotspot feature are able to provide connectivity even if networks settings are in complete mess

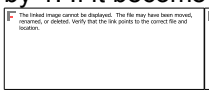
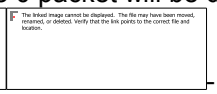

- bridge goes thought the MAC address table in order to find a match to destination MAC address of packet. When match is found - packet will be send out via corresponding bridge port. In case of no match - multiple copies of packet will be created and packet will be sent out via all bridge ports

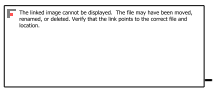

- this is a workaround, allows to use "out-bridge-port" before actual bridge decision.

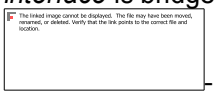

- router goes thought the route n order to find a match to destination IP address of packet. When match is found - packet will be send out via corresponding port or to the router itself . In case of no match - packet will be discarded.


- this is a workaround that allows to set-up policy routing in mangle chain output


- indicates exact place where Time To Live (TTL) of the routed packet is reduced by 1. If it become 0 packet will be discarded

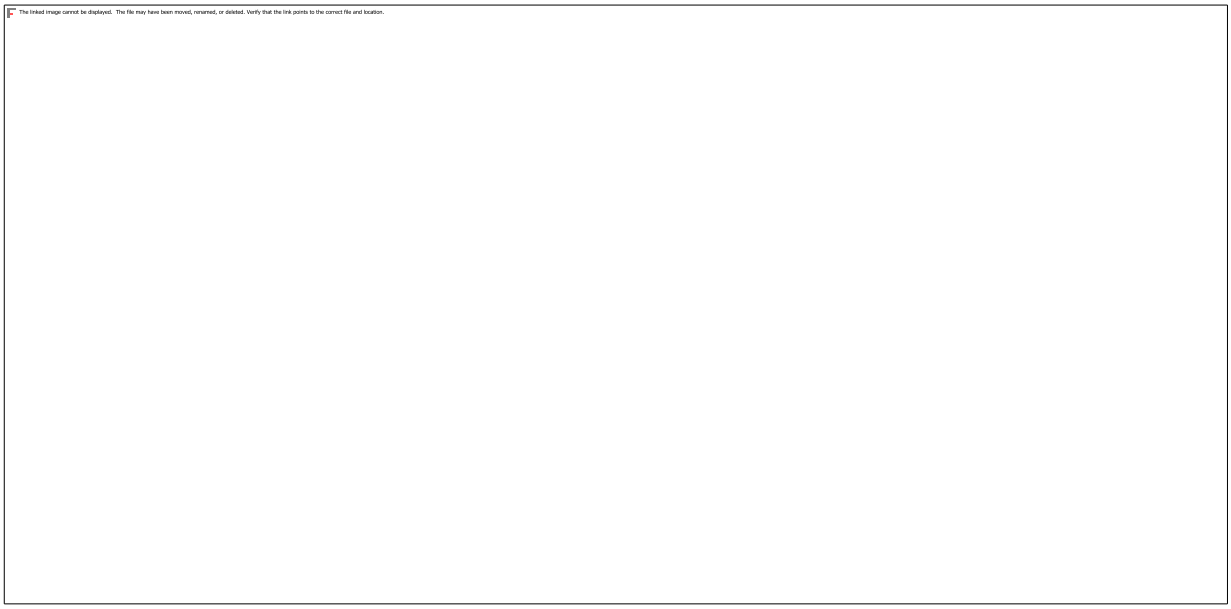


- self explainatory


- check if the *actual output interface* is a port for bridge OR checks if *output interface* is bridge

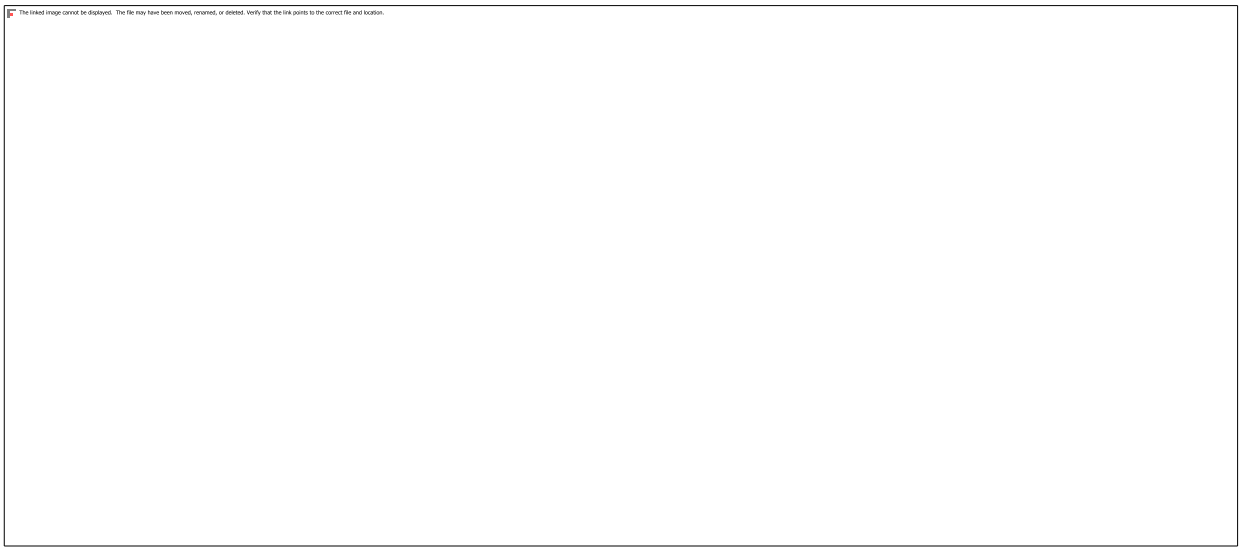

- undo all that was done by hotspot-in for the packets that is going back to client.

Examples

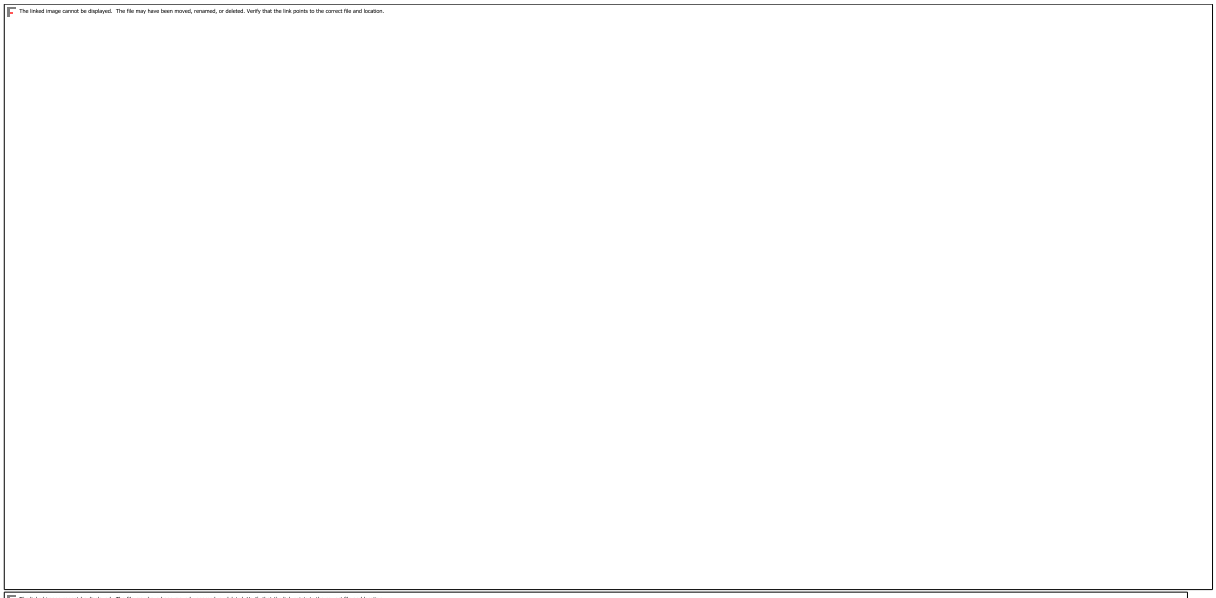
Bridging with use-ip-firewall=yes

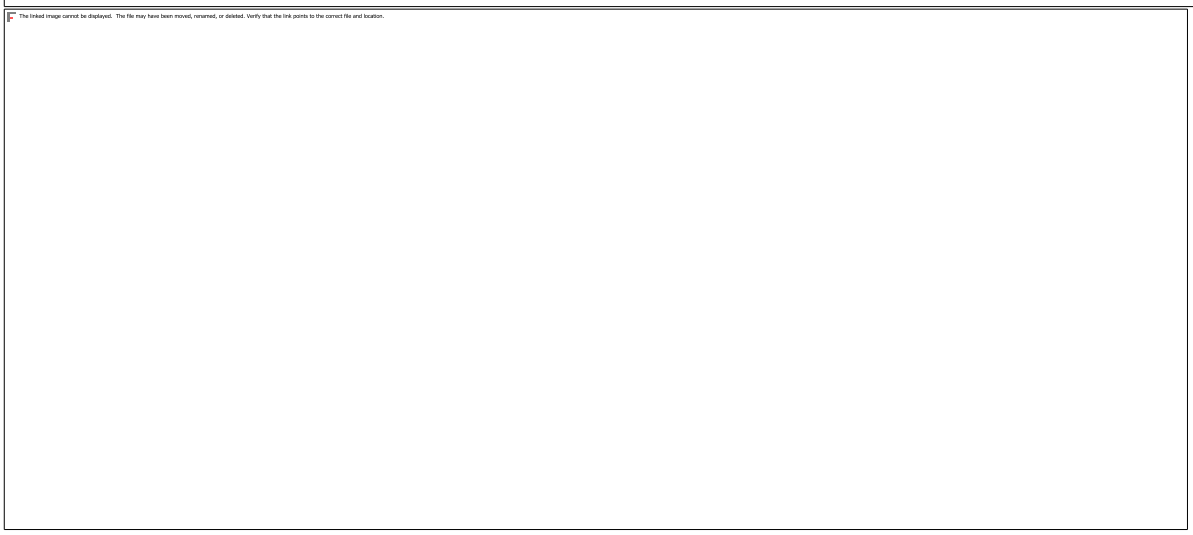

The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

Routing - from Ethernet to Ethernet interface

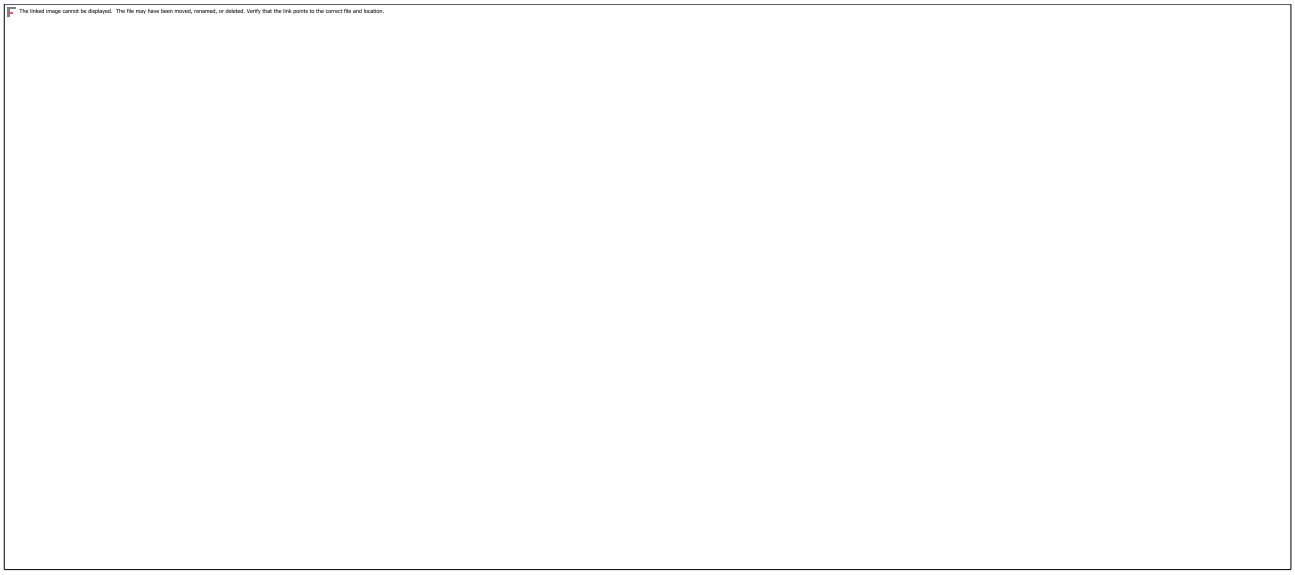

The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

Routing from one Bridge interface to different Bridge interface

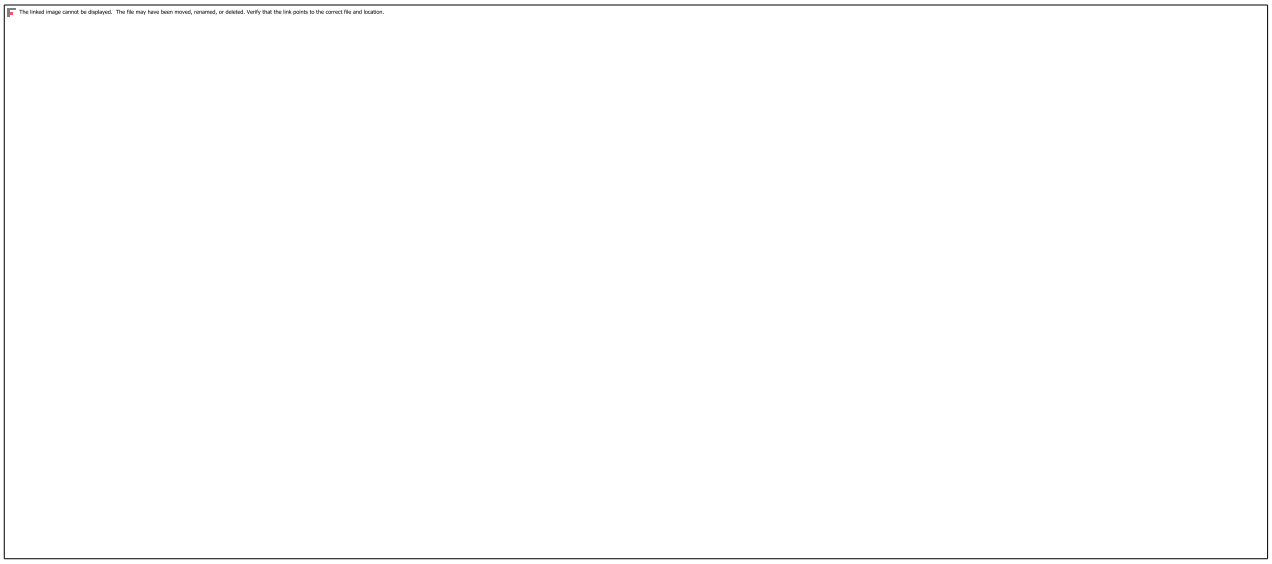

The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.


The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

IPsec encryption


The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

IPsec decryption


The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

Queues

Submenu level: **/queue**
Queues are used to limit and prioritize traffic. They can be used to

- limit data rate for certain IP addresses, subnets, protocols, ports, and other parameters
- limit peer-to-peer traffic
- prioritize some packet flows over others
- configure traffic bursts for faster web browsing
- apply different limits based on time
- share available traffic among users equally, or depending on the load of the channel

Queue implementation in HauteRouterOS is based on Hierarchical Token Bucket (HTB). HTB allows to create hierarchical queue structure and determine relations between queues.
In HauteRouterOS, these hierarchical structures can be attached at 4 different places:

- *global-in*: represents all the input interfaces in general (INGRESS queue). Queues attached to global-in apply to traffic that is received by the router before the packet filtering
- *global-out*: represents all the output interfaces in general (EGRESS queue).
- *global-total*: represents all input and output interfaces together (in other words it is aggregation of global-in and global-out). Used in case when customers have single limit for both, upload and download.
- <*interface name*>: - represents one particular outgoing interface. Only traffic that is designated to go out via this interface will pass this HTB queue.

There are two different ways how to configure queues in HauteRouterOS:

- **/queue simple** menu - designed to ease configuration of simple, everyday queuing tasks (such as single client upload/download limitation, p2p traffic limitation, etc.).
- **/queue tree** menu - for implementing advanced queuing tasks (such as global prioritization policy, user group limitations). Requires marked packet flows from **/ip firewall mangle** facility.

Simple Queues

Submenu level: **/queue simple**
One configuration item in */queue simle'* can create from 0 to 3 separate queues - one queue in *global-in*, one queue in *global-out* and one queue in *global-total*. If all properties of a queue have default values (no set limits, queue type is *default*), and queue has no children, then it is not actually created. This way, for exanple, creation of *global-total* queues can be avoided if only upload/download limitation is used.
Simple queues have strict order - each packet must go through every queue until it will meet conditions. (In case of 1000 queues, packet for last queue will need to proceed through 999 queues before it will reach the destination) {{{...}}}

Flow Identifiers

- **target-addresses** (multiple choice: IP address/netmask) : list of IP address ranges that will be limited by this queue.
- **interface** (Name of the interface, or *all*) : identifies interface the target is connected to. Useful when it is not possible to specify targets addresses.

Each of these two properties can be used to determine which direction is target upload and which is download.
Be careful to configure both of these options for the same queue - in case they will point to opposite directions queue will not work.
If neither value of **target-addresses** nor of **interface** is specified, the queue will not be able to make difference between upload and download, and will limit all traffic twice.

Other properties

- **name** (Text) : Unique queue identifier that can be used as **parent** option value for other queues
- **direction** (One of *both*, *upload*, *download*, *none*; default: *both*) : allow to enable one-directional limitation for simple queues (disable other direction)
    - *both* - limit both download and upload traffic
    - *upload* - limit only traffic to the target
    - *download* - limit only traffic from the target
- **time** (*TIME-TIME,sun,mon,tue,wed,thu,fri,sat* - *TIME* is local time, all day names are optional; default: not set) : allow to specify time when particular queue will be active. Router must have correct time settings.
- **dst-address** (IP address/netmask) : allows to select only specific stream (from target address to this destination address) for limitation explain what is target and what is dst and what is upload and what not
- **p2p** (one of *all-p2p*, *bit-torrent*, *blubster*, *direct-connect*, *edonkey*, *fasttrack*, *gnutella*, *soulseek*, *winmx*; default: not set) : allow to select unencrypted packets of particular p2p for limitation
- **packet-marks** (Comma separated list of packet mark names) : allows to use marked packets from **/ip firewall mangle**. Take look at the HauteRouterOS packet flow diagram (above). It is necessary to mark packets before the simple queues (before *global-in* HTB queue) or else target's download limitation will not work. The only mangle chain before *global-in* is *prerouting*.

HTB Properties

- **parent** (Name of parent simple queue, or *none*) : assigns this queue as a child queue for selected target {{{...}}}. Target queue can be HTB queue or any other previously created simple queue. In order for traffic to reach child queues, parent queues must capture all necessary traffic.
- **priority** (1..8) : Prioritize one child queue over other child queue. Does not work on parent queues (if queue has at least one child). One is the highest, eight is the lowest priority. Child queue with higher priority will have chance to reach its **limit-at** before child with lower priority and after that child queue with higher priority will have chance to reach its **max-limit** before child with lower priority. Priority have nothing to do with bursts.
- **queue** (*SOMETHING/SOMETHING*) : Choose the type of the upload/download queue. Queue types can be created in **/queue type**.
- **limit-at** (*NUMBER/NUMBER*) : normal upload/download data rate that is guaranteed to a target
- **max-limit** (*NUMBER/NUMBER*) : maximal upload/download data rate that is allowed for a target to reach to reach what
- **burst-limit** (*NUMBER/NUMBER*) : maximal upload/download data rate which can be reached while the burst is active
- **burst-time** (*TIME/TIME*) : period of time, in seconds, over which the average upload/download data rate is calculated. (This is NOT the time of actual burst)
- **burst-threshold** (*NUMBER/NUMBER*) : when average data rate is below this value - burst is allowed, as soon as average data rate reach this value - burst is denied. (basically this is burst on/off switch). For optimal burst behavior this value should above **limit-at** value and below **max-limit** value

And corresponding options for *global-total* HTB queue:

- **total-queue** (*SOMETHING/SOMETHING*): corresponds to **queue**
- **total-limit-at** (*NUMBER/NUMBER*): corresponds to **limit-at**
- **total-max-limit** (*NUMBER/NUMBER*): corresponds to **max-limit**
- **total-burst-limit** (*NUMBER/NUMBER*): corresponds to **burst-limit**
- **total-burst-time** (*TIME/TIME*): corresponds to **burst-time**
- **total-burst-threshold** (*NUMBER/NUMBER*): corresponds to **burst-threshold**

Good practice suggests that:

Sum of children's **limit-at** values must be less or equal to **max-limit** of the parent. Every child's **max-limit** must be less than **max-limit** of the parent. This way you will leave some traffic for the other child queues, and they will be able to get traffic without fighting for it with other child queues.

Statistics

- **rate** (read-only/read-only) : average queue passing data rate in bytes per second
- **packet-rate** (read-only/read-only) : average queue passing data rate in packets per second
- **bytes** (read-only/read-only) : number of bytes processed by this queue
- **packets** (read-only/read-only) : number of packets processed by this queue
- **queued-bytes** (read-only/read-only) : number of bytes waiting in the queue
- **queued-packets** (read-only/read-only) : number of packets waiting in the queue
- **dropped** (read-only/read-only) : number of dropped packets
- **borrows** (read-only/read-only) : packets that passed queue over its "limit-at" value (and was unused and taken away from other queues)
- **lends** (read-only/read-only) : packets that passed queue below its "limit-at" value OR if queue is a parent - sum of all child borrowed packets
- **pcq-queues** (read-only/read-only) : number of PCQ substreams, if queue type is PCQ

And corresponding options for *global-total* HTB queue:

- **total-rate** (read-only): corresponds to *rate*
- **total-packet-rate** (read-only): corresponds to *packet-rate*
- **total-bytes** (read-only): corresponds to *bytes*
- **total-packets** (read-only): corresponds to *packets*
- **total-queued-bytes** (read-only): corresponds to *queued-bytes*
- **total-queued-packets** (read-only): corresponds to *queued-packets*
- **total-dropped** (read-only): corresponds to *dropped*
- **total-lends** (read-only): corresponds to *lends*
- **total-borrows** (read-only): corresponds to *borrows*
- **total-pcq-queues** (read-only): corresponds to *pcq-queues*

Queue Tree

Submenu level: **/queue tree**

Queue tree creates only one directional queue in one of the HTBs. It is also the only way how to add queue on the separate interface. This way it is possible to ease mangle configuration - you don't need separate marks for download and upload - only upload will get to Public interface and only download will get to Private interface.

Also it is possible to have double queuing (example:prioritization of traffic in global-in or global-out, limitation per client on the outgoing interface) If you have simple queues and queue tree in the same HTB - simple queues will get traffic first.
Queue tree is not ordered - all traffic pass it together.

Flow Identifiers

- **name** (Text) : Unique queue identifier that can be used as **parent** option value for other queues
- **packet-marks** (Comma separated list of) : allows to use marked packets from **/ip firewall mangle**. Take look at this packet flow diagram (above). You need to make sure that packets are marked before the simple queues (before global-in HTB queue)

HTB Properties

- **parent** (Name of , or *none*) : assigns this queue as a child queue for selected target. Target queue can be HTB queue or any other previously created queue
- **priority** (1..8) : Prioritize one child queue over other child queue. Does not work on parent queues (if queue has at least one child). One is the highest, eight is the lowest priority. Child queue with higher priority will have chance to reach its **limit-at** before child with lower priority and after that child queue with higher priority will have chance to reach its **max-limit** before child with lower priority. Priority have nothing to do with bursts.
- **queue** (*SOMETHING*) : Choose the type of the queue. Queue types can be created here
- **limit-at** (*NUMBER*) : normal data rate that is guaranteed to a target
- **max-limit** (*NUMBER*) : maximal data rate that is allowed for a target to reach
- **burst-limit** (*NUMBER*) : maximal data rate which can be reached while the burst is active
- **burst-time** (*TIME*) : period of time, in seconds, over which the average data rate is calculated. (This is NOT the time of actual burst)
- **burst-threshold** (*NUMBER*) : when average data rate is below this value - burst is allowed, as soon as average data rate reach this value - burst is denied. (basically this is burst on/off switch). For optimal burst behavior this value should above **limit-at** value and below **max-limit** value

Statistics

- **rate** (read-only) : average queue passing data rate in bytes per second
- **packet-rate** (read-only) : average queue passing data rate in packets per second
- **bytes** (read-only) : number of bytes processed by this queue
- **packets** (read-only) : number of packets processed by this queue
- **queued-bytes** (read-only) : number of bytes waiting in the queue
- **queued-packets** (read-only) : number of packets waiting in the queue
- **dropped** (read-only) : number of dropped packets
- **borrows** (read-only) : packets that passed queue over its "limit-at" value (and was unused and taken away from other queues)

- **lends** (read-only) : packets that passed queue below its "limit-at" value OR if queue is a parent - sum of all child borrowed packets
- **pcq-queues** (read-only) : number of PCQ substreams, if queue type is PCQ

Queue Types

Submenu level: **/queue type**

- **name** (Text) : Unique queue identifier that can be used in simple queues and queue tree as value of **queue** option
- **kind** (bfifo | pcq | pfifo | red | sfq) : kind of particular queue type

PFIFO and BFIFO

These queuing disciplines are based on the FIFO algorithm (First-In First-Out). The difference between PFIFO and BFIFO is that one is measured in packets and the other one in bytes.

- **pfifo-limit** (number) : Maximum number of packets that the PFIFO queue can hold
- **bfifo-limit** (number) : Maximum number of bytes that the BFIFO queue can hold

Every packet that cannot be enqueued (if the queue is full), is dropped. Large queue sizes can increase latency, but utilize channel better.

RED

Random Early Drop is a queuing mechanism which tries to avoid network congestion by controlling the average queue size. When the average queue size reaches **red-min-threshold**, RED starts to drop packets randomly with linearly increasing probability as the average queue size grows up until the average queue size reaches the **red-max-threshold**. The effective queue size at any moment could be higher than the **red-max-threshold** as the probability does not grow very fast, so it is possible to specify a hard limit for the queue size. When the average queue size reaches red-max-threshold or becomes larger, all further packets are dropped until the average queue size does not drop below this values (at which point probabilistic calculations will be activated again).

- **red-avg-packet** (number) : Used by RED for average queue size calculations (for packet to byte translation)
- **red-burst** (number) : Number of packets allowed for bursts of packets when there are no packets in the queue
- **red-limit**(number) : RED queue limit in packets
- **red-max-threshold** (number) : The average queue size at which packet marking probability is the highest
- **red-min-threshold** (number) : Average queue size in bytes

SFQ

Stochastic Fairness Queuing (SFQ) is ensured by hashing and round-robin algorithms. A traffic flow may be uniquely identified by a 4 options(src-address, dst-address, src-port and dst-port), so these parameters are used by SFQ hashing algorithm to classify packets into one of 1024 possible sub-streams. Then round-robin algorithm will start to distribute available bandwidth to all sub-streams, on each round giving **sfq-allot** bytes of traffic. The whole SFQ queue can contain 128 packets and there are 1024 sub-streams available.

- **sfq-allot** (number) : How often hash function must be refreshed
- **sfq-perturb** (time) : Amount of data in bytes that can be sent in one round-robin round

PCQ

Per Connection Queuing (PCQ) is a similar to SFQ, but it has additional features.

It is possible to choose flow identifiers (from dst-address | dst-port | src-address | src-port). For example if you classify flows by src-address on local interface (interface with your clients), each PCQ sub-stream will be one particular client's upload.
It is possible to assign speed limitation to sub-streams with **pcq-rate** option. If **pcq-rate**=0 sub-streams will divide available traffic equally.

PCQ parameters:

- **pcq-classifier** (dst-address | dst-port | src-address | src-port; default: "") : selection of sub-stream identifiers
- **pcq-rate** (number) : maximal available data rate of each sub-steam
- **pcq-limit** (number) : queue size of one sub-stream in packets
- **pcq-total-limit** (number) : queue size of global FIFO queue

Interface Queue

Submenu level: **/queue interface**

- **interface**(*SOMETHING*) : name of interface
- **queue** (*something*) : queue type assigned to particular interface

HTB

Theory

Structure

Hierarchical Token Bucket (HTB) allows to create a hierarchical queue structure and determine relations between queues, like "parent-child" or "child-child".
As soon as queue has at least one child it becomes a **inner** queue, all queues without children - **leaf** queues. **Leaf** queues make actual traffic consumption, **Inner** queues are responsible only for traffic distribution. All **leaf** queues are treated on equal basis.
In HauteRouterOS it is necessary to specify **parent** option to assign queue as a child to other queue

Dual Limitation

Each queue in HTB has two rate limits:

- **CIR** (Committed Information Rate) – (**limit-at** in HauteRouterOS) worst case scenario, flow will get this amount of traffic no matter what (assuming we can actually send so much data)
- **MIR** (Maximal Information Rate) – (**max-limit** in HauteRouterOS) best case scenario, rate that flow can get up to, if there queue's parent has spare bandwidth

In another words, at first **limit-at** (**CIR**) of the all queues will be satisfied, only then child queues will try to borrow the necessary data rate from their parents in order to reach their **max-limit** (**MIR**).
Note: **CIR** will be assigned to the corresponding queue no matter what. (even if max-limit of the parent is exceeded)
That is why, to ensure optimal (as designed) usage of dual limitation feature, we suggest to stick to these rules:

- Sum of committed rates of all children must be less or equal to amount of traffic that is available to parent.

  CIR(parent) ≥ CIR(child1) +...+ CIR(childN)*
  **in case if parent is main parent CIR(parent)=MIR(parent)*

- Maximal rate of any child must be less or equal to maximal rate of the parent

  MIR (parent) ≥ MIR(child1) & MIR (parent) ≥ MIR(child2) & ... & MIR (parent) ≥ MIR(childN)

Queue colors in HSConfig:

- 0% - 50% available traffic used - green
- 51% - 75% available traffic used - yellow
- 76% - 100% available traffic used - red

Priority

We already know that **limit-at** (**CIR**) to all queues will be given out no matter what.
Priority is responsible for distribution of remaining parent queues traffic to child queues so that they are able to reach **max-limit**
Queue with higher priority will reach its **max-limit** before the queue with lower priority. 8 is the lowest priority, 1 is the highest.
Make a note that priority only works:

- for **leaf** queues - priority in **inner** queue have no meaning.
- if **max-limit** is specified (not 0)

Examples

In this section we will analyze HTB in action. To do that we will take one HTB structure and will try to cover all the possible situations and features, by changing the amount of incoming traffic that HTB have to recycle. and changing some options.

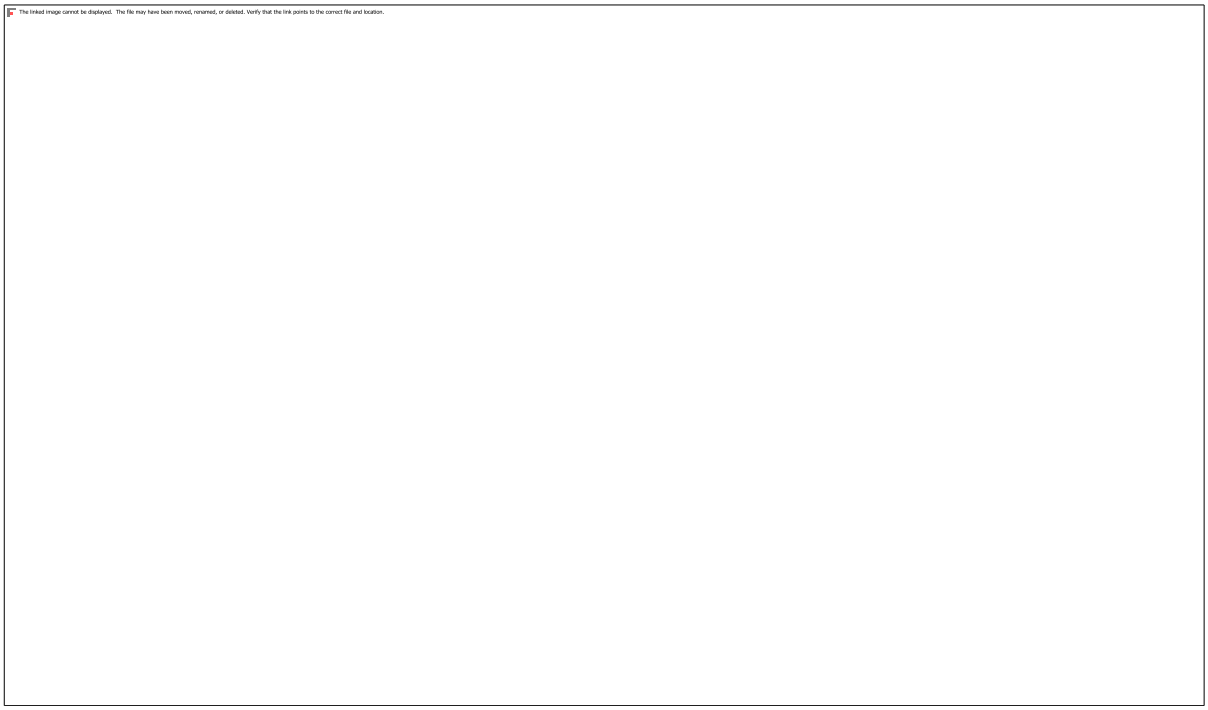
Structure

Our HTB structure will consist of 5 queues:

- **Queue01** inner queue with two children - **Queue02** and **Queue03**
- **Queue02** inner queue with two children - **Queue04** and **Queue05**
- **Queue03** leaf queue
- **Queue04** leaf queue
- **Queue05** leaf queue

Queue03, **Queue04** and **Queue05** are clients who require 10Mbps all the time Outgoing interface is able to handle 10Mbps of traffic.

Example 1 : Usual case


The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

- **Queue01** limit-at=0Mbps max-limit=10Mbps
- **Queue02** limit-at=4Mbps max-limit=10Mbps
- **Queue03** limit-at=6Mbps max-limit=10Mbps priority=1
- **Queue04** limit-at=2Mbps max-limit=10Mbps priority=3
- **Queue05** limit-at=2Mbps max-limit=10Mbps priority=5

Result of Example 1

- **Queue03** will receive 6Mbps
- **Queue04** will receive 2Mbps
- **Queue05** will receive 2Mbps
- **Clarification:** HTB was build in a way, that, by satisfying all **limit-at**s, main queue no longer have throughput to distribute

Example 2 : Usual case with max-limit

The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

- **Queue01** limit-at=0Mbps max-limit=10Mbps
- **Queue02** limit-at=4Mbps max-limit=10Mbps
- **Queue03** limit-at=2Mbps max-limit=10Mbps priority=3
- **Queue04** limit-at=2Mbps max-limit=10Mbps priority=1
- **Queue05** limit-at=2Mbps max-limit=10Mbps priority=5

Result of Example 2

- **Queue03** will receive 2Mbps
- **Queue04** will receive 6Mbps
- **Queue05** will receive 2Mbps
- **Clarification:** After satisfying all **limit-at**s HTB will give throughput to queue with highest priority.

Example 3 : Inner queue limit-at

The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

- **Queue01** limit-at=0Mbps max-limit=10Mbps
- **Queue02** limit-at=8Mbps max-limit=10Mbps
- **Queue03** limit-at=2Mbps max-limit=10Mbps priority=1
- **Queue04** limit-at=2Mbps max-limit=10Mbps priority=3
- **Queue05** limit-at=2Mbps max-limit=10Mbps priority=5

Result of Example 3

- **Queue03** will receive 2Mbps
- **Queue04** will receive 6Mbps
- **Queue05** will receive 2Mbps
- **Clarification:** After satisfying all **limit-at**s HTB will give throughput to queue with highest priority. But in this case **inner** queue **Queue02** had **limit-at** specified, by doing so, it reserved 8Mbps of throughput for queues **Queue04** and **Queue05**. From these two **Queue04** have highest priority, that is why it gets additional throughput.

Example 4 : Leaf queue limit-at

The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

- **Queue01** limit-at=0Mbps max-limit=10Mbps
- **Queue02** limit-at=4Mbps max-limit=10Mbps
- **Queue03** limit-at=6Mbps max-limit=10Mbps priority=1
- **Queue04** limit-at=2Mbps max-limit=10Mbps priority=3
- **Queue05** limit-at=12Mbps max-limit=15Mbps priority=5

Result of Example 4

- **Queue03** will receive ~3Mbps
- **Queue04** will receive ~1Mbps
- **Queue05** will receive ~6Mbps
- **Clarification:** Only by satisfying all **limit-at**s HTB was forced to allocate 20Mbps - 6Mbps to **Queue03**, 2Mbps to **Queue04**, 12Mbps to **Queue05**, but our output interface is able to handle 10Mbps. As output interface queue is usually FIFO throughput allocation will keep ratio 6:2:12 or 3:1:6

DSCP based QoS with HTB

This page describes a way to prioritize traffic by using DSCP tags. The DiffServ Code Point is a field in the IP header that allows you to classify traffic. DSCP is meant to be administered in a per-hob-based way, allowing each router on a path to determine how each traffic class should be prioritized. The solution described in this document is built around the Hierarchical Token Bucket queuing algorithm in RouterOS, dividing the 64 possible DSCP code values into the 8 queues available. This solution also utilizes the tree-based queuing, in order to have a parent queue do bandwidth control, with sub-queues for each possible DSCP value.
The actual queuing is done as per this table:

Name	Precendence	DSCP Range	HTB Priority
Routing (default)	000 (0)	000000(0) – 000111 (7)	8
Priority	001 (1)	001000 (8) – 001111 (15)	7
Immediate	010((2)	010000 (16) – 010111 (23)	6
Flash	011 (3)	011000 (24) – 011111 (31)	5
Flash Override	100 (4)	100000 (32) – 100111 (39)	4
Critical	101 (5)	101000 (40) – 101111 (47)	3
Internetwork Control	110 (6)	111000 (48) – 110111 (55)	2
Network Control	111 (7)	111000 (56) – 111111 (63)	1

DSCP marking/mangling

In order to match DSCP values in your queues, it is necessary to mark the packets using firewall mangling. This is best done with this command:

```
:for x from 0 to 63 do={/ip firewall mangle add action=mark-
packet chain=postrouting \
comment=("dscp_" . $x . "_eth") disabled=no dscp=$x new-packet-
mark=("dscp_" . $x . "_eth") passthrough=no}
```

This command creates 64 lines under /ip firewall mangle, that simply marks each packet with a DSCP value to be processed later.
Having that done, it's time to move on to the actual queues.

Setting up the queue tree

The next example assumes that ether1 is the wan interface, and your available bandwidth is 5Mbit/s.

```
/queue tree
add burst-limit=0 burst-threshold=0 burst-time=0s disabled=no
limit-at=0 max-limit=5000000 name=ether1 \
parent=ether1 queue=default

#prio8
:for z from 0 to 7 do={/queue tree add burst-limit=0 burst-
threshold=0 burst-time=0s disabled=no limit-at=0 max-limit=0 \
name=("routine_" . $z . "_ether1") packet-mark=("dscp_" . $z .
"_eth") parent=ether1 priority=8 queue=ethernet-default}

#prio7
:for z from 8 to 15 do={/queue tree add burst-limit=0 burst-
threshold=0 burst-time=0s disabled=no limit-at=0 max-limit=0 \
name=("priority_" . $z . "_ether1") packet-mark=("dscp_" . $z .
"_eth") parent=ether1 priority=7 queue=ethernet-default}
```

```
#prio 6
:for z from 16 to 23 do={/queue tree add burst-limit=0 burst-
threshold=0 burst-time=0s disabled=no limit-at=0 max-limit=0 \
name=("immediate_" . $z . "_ether1") packet-mark=("dscp_" . $z .
"_eth") parent=ether1 priority=6 queue=ethernet-default}

#prio 5
:for z from 24 to 31 do={/queue tree add burst-limit=0 burst-
threshold=0 burst-time=0s disabled=no limit-at=0 max-limit=0 \
name=("flash_" . $z . "_ether1") packet-mark=("dscp_" . $z .
"_eth") parent=ether1 priority=5 queue=ethernet-default}

#prio 4
:for z from 32 to 39 do={/queue tree add burst-limit=0 burst-
threshold=0 burst-time=0s disabled=no limit-at=0 max-limit=0 \
name=("flash_override_" . $z . "_ether1") packet-mark=("dscp_" .
$z . "_eth") parent=ether1 priority=4 queue=ethernet-default}

#prio 3
:for z from 40 to 47 do={/queue tree add burst-limit=0 burst-
threshold=0 burst-time=0s disabled=no limit-at=0 max-limit=0 \
name=("critical_" . $z . "_ether1") packet-mark=("dscp_" . $z .
"_eth") parent=ether1 priority=3 queue=ethernet-default}

#prio 2
:for z from 48 to 55 do={/queue tree add burst-limit=0 burst-
threshold=0 burst-time=0s disabled=no limit-at=0 max-limit=0 \
name=("intercon_" . $z . "_ether1") packet-mark=("dscp_" . $z .
"_eth") parent=ether1 priority=2 queue=ethernet-default}

#prio 1
:for z from 56 to 63 do={/queue tree add burst-limit=0 burst-
threshold=0 burst-time=0s disabled=no limit-at=0 max-limit=0 \
name=("netcon_" . $z . "_ether1") packet-mark=("dscp_" . $z .
"_eth") parent=ether1 priority=1 queue=ethernet-default}
```

Remarks

This solution is the most flexible solution available. It is built around the philosophy that highest DSCP marking is served first. The actual shaping of the interface could be moved to a simple queue in order to be able to police differently on upstream and downstream, but I prefer to shape in both ends of a circuit, so when you have different upload and download speed, you should shape in according to the upload speed.

HauteRouterOS Packet Flow

Document revision: 2.6 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

This manual describes the order in which an IP packet traverses various internal facilities of the router and some general information regarding packet handling, common IP protocols and protocol options.

Specifications

Packages required: ***system***
License required: *Level3*
Submenu level: ***/ip firewall***
Standards and Technologies: IP
Hardware usage: *Increases with NAT, mangle and filter rules count*

Packet Flow

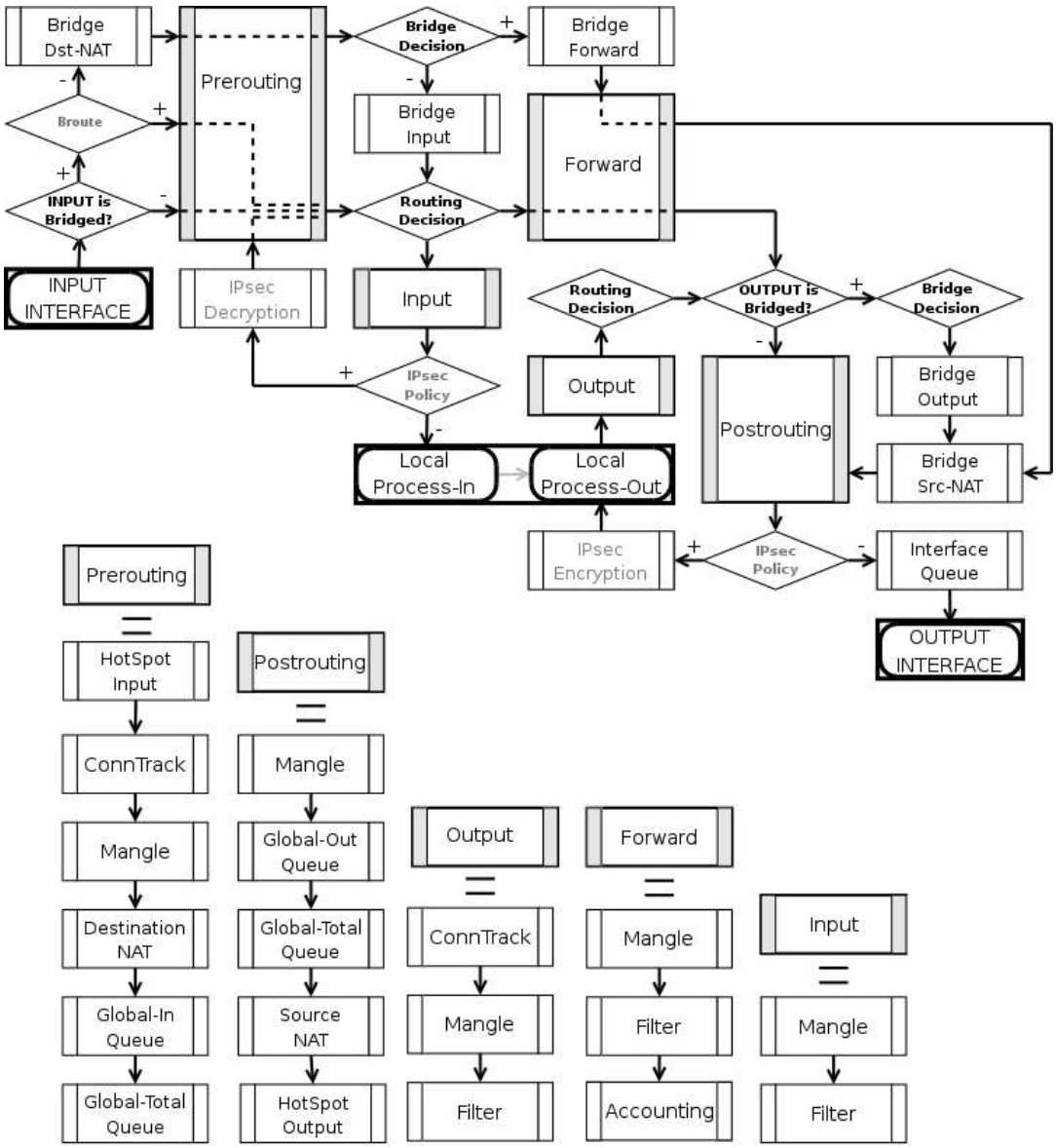
Description

Hautespot RouterOS is designed to be easy to operate in various aspects, including IP firewall. Therefore regular firewall policies can be created and deployed without the knowledge about how the packets are processed in the router. For example, if all that required is just natting internal clients to a public address, the following command can be issued (assuming the interface to the Internet in named **Public**):

```
/ip firewall nat add action=masquerade out-interface=Public chain=srcnat
```

Regular packet filtering, bandwith management or packet marking can be configured with ease in a similar manner. However, a more complicated configuration could be deployed only with a good understanding of the underlying processes in the router.

The packet flow through the router is depicted in the following diagram:



As can be seen on the diagram, there are five chains in the processing pipeline. These are **prerouting**, **input**, **forward**, **output** and **postrouting**. The actions performed on a packet in each chain are discussed later in this chapter.

A paket can enter processing conveyer of the router in two ways. First, a packet can come from one of the interfaces present in the roter (then the interface is referred as **input interface**). Second, it can be originated from a local process, like web proxy, VPN or others. Alike, there are two ways for a packet to leave the processing pipeline. A packet can leave through the one of the router's interfaces (in this case the interface is referred as **output interface**) or it can end up in the local process. In general, traffic can be destined to one of the router's IP addresses, it can originate from the router or simply should be passed through. To further complicate things the

traffic can be bridged or routed one, which is determined during the **Bridge Decision** stage.

Routed traffic

The traffic which is being routed can be one of three types:

- the traffic which is destined to the router itself. The IP packets has destination address equal to one of the router's IP addresses. A packet enters the router through the **input interface**, sequentially traverses **prerouting** and **input** chains and ends up in the local process. Consequently, a packet can be filtered in the **input** chain filter and mangled in two places: the **input** and the **prerouting** chain filters.
- the traffic originated by the router. In this case the IP packets have their source addresses identical to one of the router's IP addresses. Such packets travel through the **output** chain, then they are passed to the routing facility where an appropriate routing path for each packet is determined and leave through the **postrouting** chain.
- one which passes through the router. These packets go through the **prerouting**, **forward** and **postrouting** chains.

The actions imposed by various router facilities are sequentially applied to a packet in each of the default chains. The exact order they are applied is pictured in the bottom of the flow diagram. *Exempli gratia*, for a packet passing **postrouting** chain the mangle rules are applied first, two types of queuing come in second place and finally source NAT is performed on packets that need to be natted.

Note, that a given packet can come through only one of the **input**, **forward** or **output** chains.

Bridged Traffic

In case the incoming traffic needs to be bridged (do not confuse it with the traffic coming from the bridge interface, which should be routed) it is first determined whether it is an IP traffic or not. After that the IP traffic goes through the **prerouting**, **forward** and **postrouting** chains, while non-IP traffic goes directly to the interface queue. Both types of traffic, however, undergo the bridge firewall check first.

Additional arrows from IPsec boxes shows the processing of encrypted packets (they need to be encrypted / decrypted first and then processed as usual, *id est* from the point an ordinal packet enters the router).

If the packet is bridged one, the 'Routing Decision' changes to 'Bridge Forwarding Decision'. In case the bridge is forwarding non-IP packets, all things regarding IP protocol are not applicable ('Universal Client', 'Conntrack', 'Mangle', *et cetera*).

Connection Tracking

Submenu level: ***/ip firewall connection***

Description

Connection tracking refers to the ability to maintain the state information about connections, such as source and destination IP address and ports pairs, connection states, protocol types and timeouts. Firewalls that do connection tracking are known as "stateful" and are inherently more secure that those who do only simple "stateless" packet processing.

The *state* of a particular connection could be **estabilished** meaning that the packet is part of already known connection, **new** meaning that the packet starts a new connection or belongs to a connection that has not seen packets in both directions yet, **related** meaning that the packet starts a new connection, but is associated with an existing connection, such as FTP data transfer or ICMP error message and, finally, **invalid** meaning that the packet does not belong to any known connection.

Connection tracking is done either in the **prerouting** chain, or the **output** chain for locally generated packets.

Another function of connection tracking which cannot be overestimated is that it is needed for NAT. You should be aware that no NAT can be performed unless you have connection tracking enabled, the same applies for p2p protocols recognition. Connection tracking also assembles IP packets from fragments before further processing.

The maximum number of connections the **/ip firewall connection** state table can contain is determined initially by the amount of physical memory present in the router. Thus, for example, a router with 64 MB of RAM can hold the information about up to 65536 connections, but a router with 128 MB RAM increases this value to more than 130000.

Please ensure that your router is equipped with sufficient amount of physical memory to properly handle all connections.

Property Description

connection-mark (*read-only: text*) - Connection mark set in mangle
dst-address (*read-only: IP address:port*) - the destination address and port the connection is established to
protocol (*read-only: text*) - IP protocol name or number
p2p (*read-only: text*) - peer to peer protocol
reply-src-address (*read-only: IP address:port*) - the source address and port the reply connection is established from
reply-dst-address (*read-only: IP address:port*) - the destination address and port the reply connection is established to
src-address (*read-only: IP address:port*) - the source address and port the connection is established from
tcp-state (*read-only: text*) - the state of TCP connection

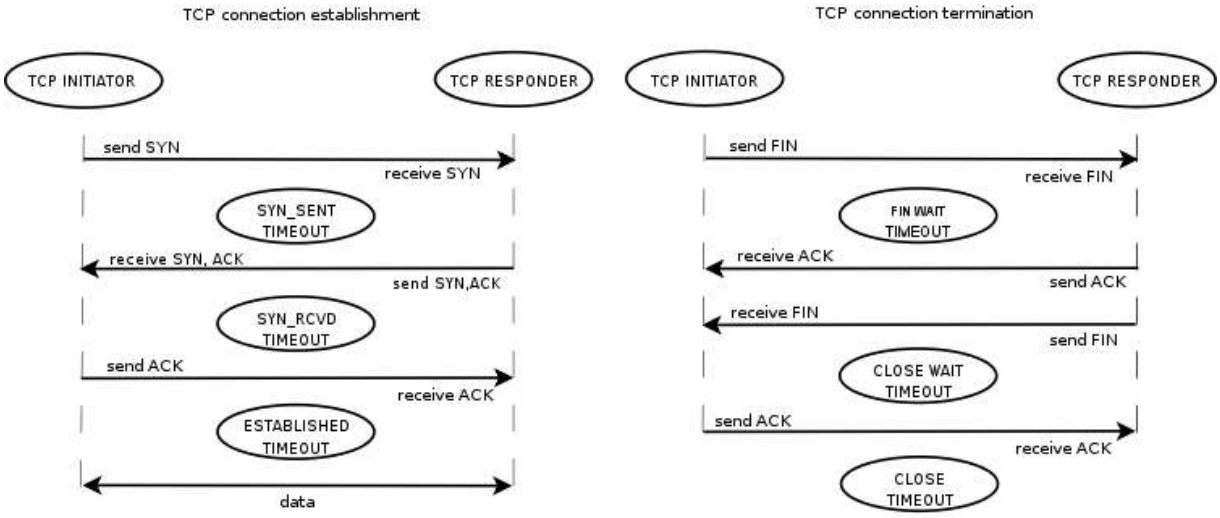
timeout (*read-only: time*) - the amount of time until the connection will be timed out
assured (*read-only:* true | false) - shows whether replay was seen for the last packet matching this entry
icmp-id (*read-only: integer*) - contains the ICMP ID. Each ICMP packet gets an ID set to it when it is sent, and when the receiver gets the ICMP message, it sets the same ID within the new ICMP message so that the sender will recognize the reply and will be able to connect it with the appropriate ICMP request
icmp-option (*read-only: integer*) - the ICMP type and code fields
reply-icmp-id (*read-only: integer*) - contains the ICMP ID of received packet
reply-icmp-option (*read-only: integer*) - the ICMP type and code fields of received packet
unreplied (*read-only:* true | false) - shows whether the request was unreplied

Connection Timeouts

Submenu level: ***/ip firewall connection tracking***

Description

Connection tracking provides several timeouts. When particular timeout expires the according entry is removed from the connection state table. The following diagram depicts typical TCP connection establishment and termination and tcp timeouts that take place during these processes:



Property Description

count-curent (*read-only: integer*) - Number of connections currently recorded in the connection state table

count-max (*read-only: integer*) - The maximum number of connections the connection state table can contain, depends on an amount of total memory
enable (yes | no; default: **yes**) - Whether to allow or disallow connection tracking
generic-timeout (*time*; default: **10m**) - Maximum amount of time connection state table entry that keeps tracking of packets that are neither TCP nor UDP (for instance GRE) will survive after having seen last packet matching this entry. Creating PPTP connection this value will be increased automaticly
icmp-timeout (*time*; default: **10s**) - Maximum amount of time connection tracking entry will survive after having seen ICMP request
tcp-close-timeout (*time*; default: **10s**) - Maximum amount of time connection tracking entry will survive after having seen connection reset request (RST) or an acknowledgment (ACK) of the connection termination request from connection release initiator
tcp-close-wait-timeout (*time*; default: **10s**) - Maximum amount of time connection tracking entry will survive after having seen an termination request (FIN) from responder
tcp-established-timeout (*time*; default: **1d**) - Maximum amount of time connection tracking entry will survive after having seen an acknowledgment (ACK) from connection initiator
tcp-fin-wait-timeout (*time*; default: **10s**) - Maximum amount of time connection tracking entry will survive after having seen connection termination request (FIN) from connection release initiator
tcp-syn-received-timeout (*time*; default: **1m**) - Maximum amount of time connection tracking entry will survive after having seen a matching connection request (SYN)
tcp-syn-sent-timeout (*time*; default: **1m**) - Maximum amount of time connection tracking entry will survive after having seen a connection request (SYN) from connection initiator
tcp-time-wait-timeout (*time*; default: **10s**) - Maximum amount of time connection tracking entry will survive after having seen connection termination request (FIN) just after connection request (SYN) or having seen another termination request (FIN) from connection release initiator
udp-timeout (*time*; default: **10s**) - Maximum amount of time connection tracking entry will survive after having seen last packet matching this entry
udp-stream-timeout (*time*; default: **3m**) - Maximum amount of time connection tracking entry will survive after replay is seen for the last packet matching this entry (connection tracking entry is assured). It is used to increase timeout for such connections as H323, VoIP, etc.

Notes

The maximum timeout value depends on amount of entries in connection state table. If amount of entries in the table is more than:

- 1/16 of maximum number of entries the maximum timeout value will be 1 day
- 3/16 of maximum number of entries the maximum timeout value will be 1 hour
- 1/2 of maximum number of entries the maximum timeout value will be 10 minute

- 13/16 of maximum number of entries the maximum timeout value will be 1 minute

If timeout value exceeds the value listed above, the less value is used

If conncection tracking timeout value is less than data packet rate, e.g. timeout expires before next packet arives, NAT and statefull-firewalling stop working

General Firewall Information

Description

ICMP TYPE:CODE values

In order to protect your router and attached private networks, you need to configure firewall to drop or reject most of ICMP traffic. However, some ICMP packets are vital to maintain network reliability or provide troubleshooting services.

The following is a list of ICMP TYPE:CODE values found in good packets. It is generally suggested to allow these types of ICMP traffic.

Ping

- **8:0** - echo request
- **0:0** - echo reply

Trace

- **11:0** - TTL exceeded
- **3:3** - Port unreachable

Path MTU discovery

- **3:4** - Fragmentation-DF-Set

General suggestion to apply ICMP filtering

- Allow ping—ICMP Echo-Request outbound and Echo-Reply messages inbound
- Allow traceroute—TTL-Exceeded and Port-Unreachable messages inbound
- Allow path MTU—ICMP Fragmentation-DF-Set messages inbound
- Block everything else

Type of Service

Internet paths vary in quality of service they provide. They can differ in cost, reliability, delay and throughput. This situation imposes some tradeoffs, *exempli gratia* the path with the lowest delay may be among the ones with the smallest

throughput. Therefore, the "optimal" path for a packet to follow through the Internet may depend on the needs of the application and its user.

As the network itself has no knowledge on how to optimize path choosing for a particular application or user, the IP protocol provides a method for upper layer protocols to convey hints to the Internet Layer about how the tradeoffs should be made for the particular packet. This method is implemented with the help of a special field in the IP protocol header, the "Type of Service" field.

The fundamental rule is that if a host makes appropriate use of the TOS facility, its network service should be at least as good as it would have been if the host had not used this facility.

Type of Service (ToS) is a standard field of IP packet and it is used by many network applications and hardware to specify how the traffic should be treated by the gateway.

Hautespot RouterOS works with the full ToS byte. It does not take account of reserverd bits in this byte (because they have been redefined many times and this approach provides more flexibility). It means that it is possible to work with DiffServ marks (Differentiated Services Codepoint, DSCP as defined in RFC2474) and ECN codepoints (Explicit Congestion Notification, ECN as defined in RFC3168), which are using the same field in the IP protocol header. Note that it does not mean that RouterOS supports DiffServ or ECN, it is just possible to access and change the marks used by these protocols.

RFC1349 defines these standard values:

- **normal** - normal service (ToS=0)
- **low-cost** - minimize monetary cost (ToS=2)
- **max-reliability** - maximize reliability (ToS=4)
- **max-throughput** - maximize throughput (ToS=8)
- **low-delay** - minimize delay (ToS=16)

Peer-to-Peer protocol filtering

Peer-to-peer protocols also known as *p2p* provide means for direct distributed data transfer between individual network hosts. While this technology powers many brilliant applications (like Skype), it is widely abused for unlicensed software and media destribution. Even when it is used for legal purposes, p2p may heavily disturb other network traffic, such as http and e-mail. RouterOS is able to recognize connections of the most popular P2P protocols and filter or enforce QOS on them.

The protocols which can be detected, are:

- **Fasttrack** (Kazaa, KazaaLite, Diet Kazaa, Grokster, iMesh, giFT, Poisoned, mlMac)
- **Gnutella** (Shareaza, XoLoX, , Gnucleus, BearShare, LimeWire (java), Morpheus, Phex, Swapper, Gtk-Gnutella (linux), Mutella (linux), Qtella (linux),

MLDonkey, Acquisition (Mac OS), Poisoned, Swapper, Shareaza, XoloX, mlMac)
- **Gnutella2** (Shareaza, MLDonkey, Gnucleus, Morpheus, Adagio, mlMac)
- **DirectConnect** (DirectConnect (AKA DC++), MLDonkey, NeoModus Direct Connect, BCDC++, CZDC++)
- **eDonkey** (eDonkey2000, eMule, xMule (linux), Shareaza, MLDonkey, mlMac, Overnet)
- **Soulseek** (Soulseek, MLDonkey)
- **BitTorrent** (BitTorrent, BitTorrent++, Shareaza, MLDonkey, ABC, Azureus, BitAnarch, SimpleBT, BitTorrent.Net, mlMac)
- **Blubster** (Blubster, Piolet)
- **WPNP** (WinMX)
- **Warez** (Warez, Ares; starting from 2.8.18) - this protocol can only be dropped, speed limiting is impossible

HauteRouterOS Packet Packing

Document revision: 0.3.0 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

The Hautespot Packet Packer Protocol optimizes the data rate usage of links using protocols that have a high overhead per packet transmitted. The basic purpose of this protocol is to better enable wireless networks to transport VoIP traffic and other traffic that uses small packet sizes of around 100 bytes.

M3P features:

- enabled by a per interface setting
- other routers with Hautespot Discovery Protocol enabled will broadcast M3P settings
- significantly increases bandwidth availability over some wireless links by approximately four times
- offer configuration settings to customize this feature

Specifications

Packages required: ***system***
License required: *Level1*
Submenu level: ***/ip packing***

Standards and Technologies: M3P
Hardware usage: *Not significant*

Description

The wireless protocol IEEE 802.11 and, to a lesser extent, Ethernet protocol have a high overhead per packet as for each packet it is necessary to access the media, check for errors, resend in case of errors occured, and send network maintenance messages (network maintenance is applicable only for wireless). The Hautespot Packet Packer Protocol improves network performance by aggregating many small packets into a big packet, thereby minimizing the network per packet overhead cost. The M3P is very effective when the average packet size is 50-300 bytes the common size of VoIP packets.

Features:

- may work on any Ethernet-like media
- is disabled by default for all interfaces
- when older version on the RouterOS are upgraded from a version without M3P to a version with discovery, current wireless interfaces will not be automatically enabled for M3P
- small packets going to the same MAC level destination (regardless of IP destination) are collected according to the set configuration and aggregated into a large packet according to the set size
- the packet is sent as soon as the maximum aggregated-packet packet size is reached or a maximum time of 15ms (+/-5ms)

Setup

Submenu level: ***/ip packing***

Description

M3P is working only between Hautespot routers, which are discovered with Hautespot Neighbor Discovery Protocol (MNDP). When M3P is enabled router needs to know which of its neighbouring hosts have enabled M3P. MNDP is used to negotiate unpacking settings of neighbours, therefore it has to be enabled on interfaces you wish to enable M3P. Consult MNDP manual on how to do it.

Property Description

aggregated-size (*integer*; default: **1500**) - the maximum aggregated packet's size
interface (*name*) - interface to enable M3P on
packing (none | simple | compress-all | compress-headers; default: **simple**) - specifies the packing mode
none - no packing is applied to packets
simple - aggregate many small packets into one large packet, minimizing network overhead per packet

compress-headers - further increase network performance by compressing IP packet header (consumes more CPU resources)
compress-all - increase network performance even more by using header and data compression (extensive CPU usage)
unpacking (none | simple | compress-all | compress-headers; default: **simple**) - specifies the unpacking mode
none - accept only usual packets
simple - accept usual packets and aggregated packets without compression
compress-headers - accept all packets except those with payload compression
compress-all - accept all packets

Notes

Level of packet compression increases like this: **none -> simple -> compress-headers -> compress-all**.

When router has to send a packet it choses minimum level of packet compression from what its own **packing** type is set and what other router's **unpacking** type is set. Same is with **aggregated-size** setting - minimum value of both ends is actual maximum size of aggregated packet used.

aggregated-size can be bigger than interface MTU if network device allows it to be (i.e., it supports sending and receiving frames bigger than 1514 bytes)

Example

To enable maximal compression on the **ether1** interface:

```
[admin@Hautespot] ip packing> add interface=ether1 packing=compress-all \
\... unpacking=compress-all
[admin@Hautespot] ip packing> print
Flags: X - disabled
  #   INTERFACE PACKING          UNPACKING        AGGREGATED-SIZE
  0   ether1    compress-all     compress-all     1500

[admin@Hautespot] ip packing>
```

HauteRouterOS Packet Sniffer

Document revision: 1.5 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

Packet sniffer is a feature that catches all the data travelling over the network, that it is able to get (when using switched network, a computer may catch only the data addressed to it or is forwarded through it).

Specifications

Packages required: ***system***
License required: *Level1*
Submenu level: ***/tool sniffer***
Standards and Technologies: none
Hardware usage: *Not significant*

Description

It allows you to "sniff" packets going through the router (and any other traffic that gets to the router, when there is no switching in the network) and view them using specific software.

Packet Sniffer Configuration

Submenu level: ***/tool sniffer***

Property Description

interface (name | all; default: **all**) - the name of the interface that receives the packets
only-headers (yes | no; default: **no**) - whether to save in the memory packets' headers only (not the whole packet)
memory-limit (*integer*; default: **10**) - maximum amount of memory to use. Sniffer will stop after this limit is reached
file-name (*text*; default: **""**) - the name of the file where the sniffed packets will be saved to
file-limit (*integer*; default: **10**) - the limit of the file in KB. Sniffer will stop after this limit is reached
streaming-enabled (yes | no; default: **no**) - whether to send sniffed packets to a remote server
streaming-server (*IP address*; default: **0.0.0.0**) - Tazmen Sniffer Protocol (TZSP) stream receiver
filter-stream (yes | no; default: **yes**) - whether to ignore sniffed packets that are destined to the stream server
filter-protocol (all-frames | ip-only | mac-only-no-ip; default: **ip-only**) - specific protocol group to filter
all-frames - sniff all packets
ip-only - sniff IP packets only
mac-only-no-ip - sniff non-IP packets only
filter-address1 (*IP address mask:port*; default: **0.0.0.0/0:0-65535**) - criterion of choosing the packets to process

filter-address2 (*IP address mask:port*; default: **0.0.0.0/0:0-65535**) - criterion of choosing the packets to process
running (yes | no; default: **no**) - if the sniffer is started then the value is **yes** otherwise **no**

Notes

filter-address1 and **filter-address2** are used to specify the two participients in communication (i.e. they will match only in the case if one of them matches the source address and the other one matches the destination address of a packet). These properties are taken in account only if **filter-protocol** is **ip-only**.

Not only **Ethernal** (http://www.ethereal.com) and **Packetyzer** (http://www.packetyzer.com) can receive the sniffer's stream but also Hautespot's program **trafr** that runs on any IA32 Linux computer and saves received packets **libpcap** file format.

Example

In the following example **streaming-server** will be added, streaming will be enabled, **file-name** will be set to *test* and packet sniffer will be started and stopped after some time:

```
[admin@Hautespot] tool sniffer>set streaming-server=10.0.0.241 \
\... streaming-enabled=yes file-name=test
[admin@Hautespot] tool sniffer> prin
            interface: all
         only-headers: no
         memory-limit: 10
            file-name: "test"
           file-limit: 10
    streaming-enabled: yes
     streaming-server: 10.0.0.241
        filter-stream: yes
      filter-protocol: ip-only
      filter-address1: 0.0.0.0/0:0-65535
      filter-address2: 0.0.0.0/0:0-65535
              running: no
[admin@Hautespot] tool sniffer>start
[admin@Hautespot] tool sniffer>stop
```

Running Packet Sniffer

Specifications

Command name: ***/tool sniffer start***, ***/tool sniffer stop***, ***/tool sniffer save***

Description

The commands are used to control runtime operation of the packet sniffer. The **start** command is used to start/reset sniffering, **stop** - stops sniffering. To save currently sniffed packets in a specific file **save** command is used.

Example

In the following example the packet sniffer will be started and after some time - stopped:

```
[admin@Hautespot] tool sniffer> start
[admin@Hautespot] tool sniffer> stop
```

Below the sniffed packets will be saved in the file named *test*:

```
[admin@Hautespot] tool sniffer> save file-name=test
[admin@Hautespot] tool sniffer> /file print
  # NAME                          TYPE         SIZE       CREATION-TIME
  0 test                          unknown      1350       apr/07/2003
16:01:52

[admin@Hautespot] tool sniffer>
```

Sniffed Packets

Submenu level: ***/tool sniffer packet***

Description

The submenu allows to see the list of sniffed packets.

Property Description

data (*read-only: text*) - specified data inclusion in packets
dst-address (*read-only: IP address*) - IP destination address
fragment-offset (*read-only: integer*) - IP fragment offset
identification (*read-only: integer*) - IP identification
ip-header-size (*read-only: integer*) - the size of IP header
ip-packet-size (*read-only: integer*) - the size of IP packet
ip-protocol (ip | icmp | igmp | ggp | ipencap | st | tcp | egp | pup | udp | hmp | xns-idp | rdp | iso-tp4 | xtp | ddp | idrp-cmtp | gre | esp | ah | rspf | vmtp | ospf | ipip | encap) - the name/number of IP protocol
ip - Internet Protocol
icmp - Internet Control Message Protocol
igmp - Internet Group Management Protocol
ggp - Gateway-Gateway Protocol
ipencap - IP Encapsulated in IP
st - st datagram mode

tcp - Transmission Control Protocol
egp - Exterior Gateway Protocol
pup - Parc Universal packet Protocol
udp - User Datagram Protocol
hmp - Host Monitoring Protocol
xns-idp - Xerox ns idp
rdp - Reliable Datagram Protocol
iso-tp4 - ISO Transport Protocol class 4
xtp - Xpress Transfer Protocol
ddp - Datagram Delivery Protocol
idpr-cmtp - idpr Control Message Transport
gre - General Routing Encapsulation
esp - IPsec ESP protocol
ah - IPsec AH protocol
rspf - Radio Shortest Path First
vmtp - Versatile Message Transport Protocol
ospf - Open Shortest Path First
ipip - IP encapsulation (protocol 4)
encap - IP encapsulation (protocol 98)
protocol (*read-only:* ip | arp | rarp | ipx | ipv6) - the name/number of ethernet protocol
ip - Internet Protocol
arp - Address Resolution Protocol
rarp - Reverse Address Resolution Protocol
ipx - Internet Packet exchange protocol
ipv6 - Internet Protocol next generation
size (*read-only: integer*) - size of packet
src-address (*IP address*) - source address
time (*read-only: time*) - time when packet arrived
tos (*read-only: integer*) - IP Type Of Service
ttl (*read-only: integer*) - IP Time To Live

Example

In the example below it's seen, how to get the list of sniffed packets:

```
[admin@Hautespot] tool sniffer packet> pr
  # TIME    INTERFACE SRC-ADDRESS             DST-ADDRESS            IP-..
SIZE
  0 0.12    ether1    10.0.0.241:1839         10.0.0.181:23 (telnet) tcp   46
  1 0.12    ether1    10.0.0.241:1839         10.0.0.181:23 (telnet) tcp   40
  2 0.12    ether1    10.0.0.181:23 (telnet)  10.0.0.241:1839        tcp   78
  3 0.292   ether1    10.0.0.181              10.0.0.4               gre   88
  4 0.32    ether1    10.0.0.241:1839         10.0.0.181:23 (telnet) tcp   40
  5 0.744   ether1    10.0.0.144:2265         10.0.0.181:22 (ssh)    tcp   76
  6 0.744   ether1    10.0.0.144:2265         10.0.0.181:22 (ssh)    tcp   76
  7 0.744   ether1    10.0.0.181:22 (ssh)     10.0.0.144:2265        tcp   40
  8 0.744   ether1    10.0.0.181:22 (ssh)     10.0.0.144:2265        tcp   76
-- more
```

Packet Sniffer Protocols

Submenu level: ***/tool sniffer protocol***

Description

In this submenu you can see all kind of protocols that have been sniffed.

Property Description

bytes (*integer*) - total number of data bytes
protocol (*read-only:* ip | arp | rarp | ipx | ipv6) - the name/number of ethernet protocol
ip - Internet Protocol
arp - Address Resolution Protocol
rarp - Reverse Address Resolution Protocol
ipx - Internet Packet exchange protocol
ipv6 - Internet Protocol next generation
ip-protocol (ip | icmp | igmp | ggp | ipencap | st | tcp | egp | pup | udp | hmp | xns-idp | rdp | iso-tp4 | xtp | ddp | idrp-cmtp | gre | esp | ah | rspf | vmtp | ospf | ipip | encap) - the name/number of IP protocol
ip - Internet Protocol
icmp - Internet Control Message Protocol
igmp - Internet Group Management Protocol
ggp - Gateway-Gateway Protocol
ipencap - IP Encapsulated in IP
st - st datagram mode
tcp - Transmission Control Protocol
egp - Exterior Gateway Protocol
pup - Parc Universal packet Protocol
udp - User Datagram Protocol
hmp - Host Monitoring Protocol
xns-idp - Xerox ns idp
rdp - Reliable Datagram Protocol
iso-tp4 - ISO Transport Protocol class 4
xtp - Xpress Transfer Protocol
ddp - Datagram Delivery Protocol
idpr-cmtp - idpr Control Message Transport
gre - General Routing Encapsulation
esp - IPsec ESP protocol
ah - IPsec AH protocol
rspf - Radio Shortest Path First
vmtp - Versatile Message Transport Protocol
ospf - Open Shortest Path First
ipip - IP encapsulation
encap - IP encapsulation
packets (*integer*) - the number of packets
port (*name*) - the port of TCP/UDP protocol
share (*integer*) - specific type of traffic compared to all traffic in bytes

Example

```
[admin@Hautespot] tool sniffer protocol> print
  # PROTOCOL IP-PR... PORT          PACKETS   BYTES   SHARE
  0 ip                              77        4592    100 %
  1 ip       tcp                    74        4328    94.25 %
  2 ip       gre                    3         264     5.74 %
  3 ip       tcp      22 (ssh)      49        3220    70.12 %
```

```
  4 ip       tcp      23 (telnet)   25        1108    24.12 %

[admin@Hautespot] tool sniffer protocol>
```

Packet Sniffer Host

Submenu level: ***/tool sniffer host***

Description

The submenu shows the list of hosts that were participating in data excange you've sniffed.

Property Description

address (*read-only: IP address*) - IP address of the host
peek-rate (*read-only: integer/integer*) - the maximum data-rate received/transmitted
rate (*read-only:* integer / integer) - current data-rate received/transmitted
total (*read-only:* integer / integer) - total packets received/transmitted

Example

In the following example we'll see the list of hosts:

```
[admin@Hautespot] tool sniffer host> print
  # ADDRESS       RATE          PEEK-RATE           TOTAL
  0 10.0.0.4      0bps/0bps     704bps/0bps         264/0
  1 10.0.0.144    0bps/0bps     6.24kbps/12.2kbps   1092/2128
  2 10.0.0.181    0bps/0bps     12.2kbps/6.24kbps   2994/1598
  3 10.0.0.241    0bps/0bps     1.31kbps/4.85kbps   242/866

[admin@Hautespot] tool sniffer host>
```

Packet Sniffer Connections

Submenu level: ***/tool sniffer connection***

Description

Here you can get a list of the connections that have been watched during the sniffing time.

Property Description

active (*read-only:* yes | no) - if **yes** the find active connections
bytes (*read-only: integer*) - bytes in the current connection
dst-address (*read-only: IP address*) - destination address
mss (*read-only: integer*) - Maximum Segment Size
resends (*read-only: integer*) - the number of packets resends in the current connection
src-address (*read-only: IP address*) - source address

Example

The example shows how to get the list of connections:

```
[admin@Hautespot] tool sniffer connection> print
Flags: A - active
  #   SRC-ADDRESS       DST-ADDRESS             BYTES     RESENDS   MSS
  0 A 10.0.0.241:1839   10.0.0.181:23 (telnet)  6/42      60/0      0/0
  1 A 10.0.0.144:2265   10.0.0.181:22 (ssh)     504/252   504/0     0/0

[admin@Hautespot] tool sniffer connection>
```

Sniff MAC Address

You can also see the source and destination MAC Addresses. To do so, at first stop the sniffer if it is running, and select a specific interface:

```
[admin@Hautespot] tool sniffer> stop
[admin@Hautespot] tool sniffer> set interface=bridge1
[admin@Hautespot] tool sniffer> start
[admin@Hautespot] tool sniffer> print
            interface: bridge1
         only-headers: no
         memory-limit: 10
            file-name:
           file-limit: 10
    streaming-enabled: no
     streaming-server: 0.0.0.0
        filter-stream: yes
      filter-protocol: ip-only
      filter-address1: 0.0.0.0/0:0-65535
      filter-address2: 0.0.0.0/0:0-65535
              running: yes
[admin@Hautespot] tool sniffer>
```

Now you have the source and destination MAC Addresses:

```
[admin@Hautespot] tool sniffer packet> print detail
 0 time=0 src-mac-address=00:0C:42:03:02:C7 dst-mac-address=00:30:4F:08:3A:E7
   interface=bridge1 src-address=10.5.8.104:1125
   dst-address=10.1.0.172:3987 (HSConfig-tls) protocol=ip ip-protocol=tcp
   size=146 ip-packet-size=146 ip-header-size=20 tos=0 identification=5088
```

```
   fragment-offset=0 ttl=126

 1 time=0 src-mac-address=00:30:4F:08:3A:E7 dst-mac-address=00:0C:42:03:02:C7
   interface=bridge1 src-address=10.1.0.172:3987 (HSConfig-tls)
   dst-address=10.5.8.104:1125 protocol=ip ip-protocol=tcp size=253
   ip-packet-size=253 ip-header-size=20 tos=0 identification=41744
   fragment-offset=0 ttl=64

 2 time=0.071 src-mac-address=00:0C:42:03:02:C7
   dst-mac-address=00:30:4F:08:3A:E7 interface=bridge1
   src-address=10.5.8.104:1125 dst-address=10.1.0.172:3987 (HSConfig-tls)
   protocol=ip ip-protocol=tcp size=40 ip-packet-size=40 ip-header-size=20
   tos=0 identification=5089 fragment-offset=0 ttl=126

 3 time=0.071 src-mac-address=00:30:4F:08:3A:E7
   dst-mac-address=00:0C:42:03:02:C7 interface=bridge1
   src-address=10.1.0.172:3987 (HSConfig-tls) dst-address=10.5.8.104:1125
   protocol=ip ip-protocol=tcp size=213 ip-packet-size=213 ip-header-size=20
   tos=0 identification=41745 fragment-offset=0 ttl=64

-- [Q quit|D dump|down]
```

HauteRouterOS Performance Graphing

Document revision: 1.0 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

Graphing is a tool which is used for monitoring various RouterOS parameters over a period of time.

Specifications

Packages required: ***system***
License required: *Level1*
Submenu level: ***/tool graphing***
Hardware usage: *Not significant*

Description

The Graphing tool can display graphics for:

- HauteWRAP health (voltage and temperature)
- Resource usage (CPU, Memory and Disk usage)
- Traffic which is passed through interfaces
- Traffic which is passed through simple queues

Graphing consists of two parts - first part collects information and other part displays data in a Web page. To access the graphics, type **http://[Router_IP_address]/graphs/** and choose a graphic to display in your Web browser.

Data from the router is gathered every 5 minutes, but saved on the system drive every **store-every** time. After rebooting the router, graphing will display information that was last time saved on the disk before the reboot.

RouterOS generates four graphics for each item:

- "Daily" Graph (5 Minute Average)
- "Weekly" Graph (30 Minute Average)
- "Monthly" Graph (2 Hour Average)
- "Yearly" Graph (1 Day Average)

To access each graphic from a network, specify this network in **allow-address** parameter for the respective item.

General Options

Submenu level: ***/tool graphing***

Property Description

store-every (5min | hour | 24hours; default: **5min**) - how often to store information on system drive

Example

To store information on system drive every hour:

```
/tool graphing set store-every=hour
[admin@Hautespot] tool graphing> print
    store-every: hour
[admin@Hautespot] tool graphing>
```

Health Graphing

Submenu level: ***/tool graphing health***

Description

This submenu provides information about HR-WRAPDX's 'health' - voltage and temperature.

Property Description

allow-address (*IP address/netmask*; default: **0.0.0.0/0**) - network which is allowed to view graphs of router health
store-on-disk (yes | no; default: **yes**) - whether to store information about traffic on system drive or not. If not, the information will be stored in RAM and will be lost after a reboot

Interface Graphing

Submenu level: ***/tool graphing interface***

Description

Shows how much traffic is passed through an interface over a period of time.

Property Description

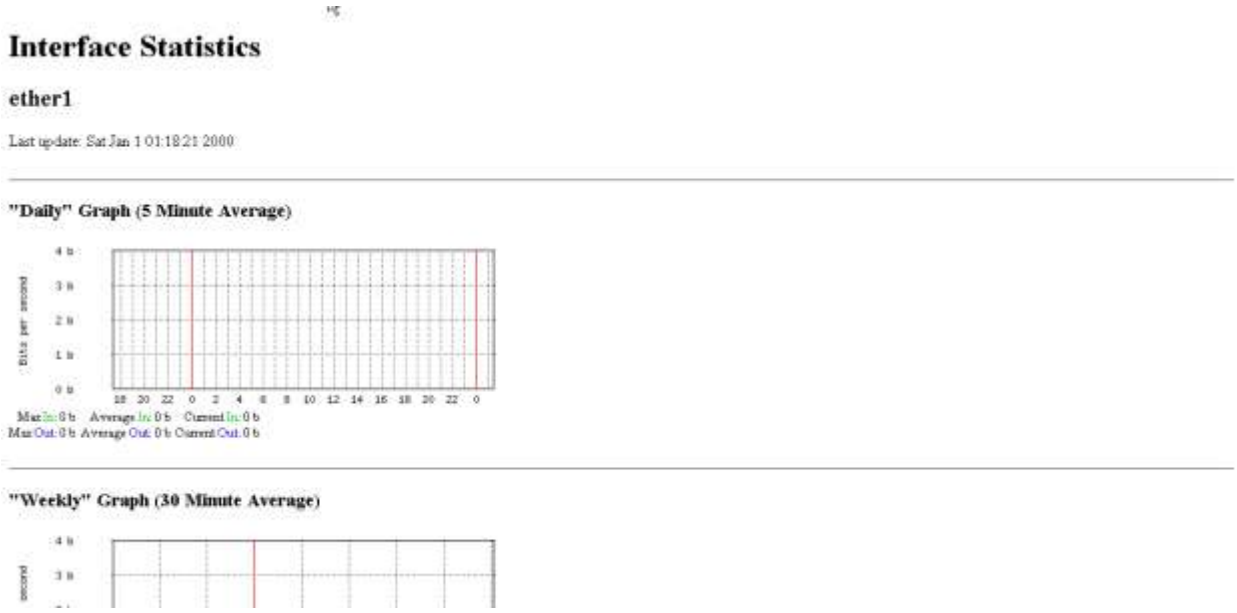
allow-address (*IP address/netmask*; default: **0.0.0.0/0**) - IP address range which is allowed to view information about the interface. If a client PC not belonging to this IP address range tries to open **http://[Router_IP_address]/graphs/**, it will not see this entry
interface (*name*; default: **all**) - name of the interface which will be monitored
store-on-disk (yes | no; default: **yes**) - whether to store information about traffic on system drive or not. If not, the information will be stored in RAM and will be lost after a reboot

Example

To monitor traffic which is passed through interface **ether1** only from local network **192.168.0.0/24**, and write information on disk:

```
[admin@Hautespot] tool graphing interface> add interface=ether1 \
\... allow-address=192.168.0.0/24 store-on-disk=yes
[admin@Hautespot] tool graphing interface> print
Flags: X - disabled
 #   INTERFACE ALLOW-ADDRESS      STORE-ON-DISK
 0   ether1    192.168.0.0/24     yes
[admin@Hautespot] tool graphing interface>
```

Graph for interface **ether1**:


Interface Statistics
ether1
Last update: Sat Jan 1 01:18:21 2000
"Daily" Graph (5 Minute Average)
"Weekly" Graph (30 Minute Average)

Simple Queue Graphing

Submenu level: ***/tool graphing queue***

Description

In this submenu you can specify a queue from the **/queue simple** list to make a graphic for it.

Property Description

allow-address (*IP address/netmask*; default: **0.0.0.0/0**) - IP address range which is allowed to view information about the queue. If a client PC not belonging to this IP address range tries to open **http://[Router_IP_address]/graphs/**, it will not see this entry
allow-target (yes | no; default: **yes**) - whether to allow access to web graphing from IP range that is specified in **/queue simple target-address**
simple-queue (*name*; default: **all**) - name of simple queue which will be monitored
store-on-disk (yes | no; default: **yes**) - whether to store information about traffic on hard drive or not. If not, the information will be stored in RAM and will be lost after a reboot

Example

Add a simple queue to Grapher list with simple-queue name **queue1**, allow limited clients to access Grapher from web, store information about traffic on disk:

```
[admin@Hautespot] tool graphing queue> add simple-queue=queue1 allow-
address=yes \
\... store-on-disk=yes
```

Resource Graphing

Submenu level: ***/tool graphing resource***

Description

Provides with router resource usage information over a period of time:

- CPU usage
- Memory usage
- Disk usage

Property Description

allow-address (*IP address*/*netmask*; default: **0.0.0.0/0**) - IP address range which is allowed to view information about the resource usage. If a client PC not belonging to this IP address range tries to open **http://[Router_IP_address]/graphs/**, it will not see this entry
store-on-disk (yes | no; default: **yes**) - whether to store information about traffic on hard drive or not. If not, the information will be stored in RAM and will be lost after a reboot

Example

Add IP range **192.168.0.0/24** from which users are allowed to monitor Grapher's resource usage:

```
[admin@Hautespot] tool graphing resource> add allow-address=192.168.0.0/24 \
\... store-on-disk=yes
[admin@Hautespot] tool graphing resource> print
Flags: X - disabled
 #   ALLOW-ADDRESS      STORE-ON-DISK
 0   192.168.0.0/24     yes
[admin@Hautespot] tool graphing resource>
```

HauteRouterOS PPP and Asynchronous Interfaces

Document revision: 1.1 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

PPP (Point-to-Point Protocol) provides a method for transmitting datagrams over serial point-to-point links. Physically it relies on **com1** and **com2** ports from standard PC hardware configurations. These appear as **serial0** and **serial1** automatically. You can add more serial ports to use the router for a modem pool using these adapters:

- MOXA (http://www.moxa.com) Smartio CP-132 2-port PCI multiport asynchronous board with maximum of 8 ports (4 cards)
- MOXA (http://www.moxa.com) Smartio C104H, CP-114 or CT-114 4-port PCI multiport asynchronous board with maximum of 16 ports (4 cards)
- MOXA (http://www.moxa.com) Smartio C168H, CP-168H or CP-168U 8-port PCI multiport asynchronous board with maximum of 32 ports (4 cards)
- Cyclades (http://www.cyclades.com) Cyclom-Y Series 4 to 32 port PCI multiport asynchronous board with maximum of 128 ports (4 cards)
- Cyclades (http://www.cyclades.com) Cyclades-Z Series 16 to 64 port PCI multiport asynchronous board with maximum of 256 ports (4 cards)
- TCL (http://www.thetcl.com) DataBooster 4 or 8 port High Speed Buffered PCI Communication Controllers

Specifications

Packages required: ***ppp***
License required: *Level1*
Submenu level: ***/interface ppp-client***, ***/interface ppp-server***
Standards and Technologies: PPP (RFC 1661)
Hardware usage: *Not significant*

Additional Resources

- http://www.ietf.org/rfc/rfc2138.txt?number=2138
- http://www.ietf.org/rfc/rfc2138.txt?number=2139

Serial Port Configuration

Submenu level: ***/port***

Property Description

name (*name*; default: **serialN**) - port name
used-by (*read-only: text*) - shows the user of the port. Only free ports can be used in PPP setup
baud-rate (*integer*; default: **9600**) - maximal data rate of the port
data-bits (7 | 8; default: **8**) - number of bits per character transmitted
parity (none | even | odd; default: **none**) - character parity check method
stop-bits (1 | 2; default: **1**) - number of stop bits after each character transmitted
flow-control (none | hardware | xon-xoff; default: **hardware**) - flow control method

Notes

Keep in mind that **baud-rate**, **data-bits**, **parity**, **stop-bits** and **flow control** parameters must be the same for both communicating sides.

Example

```
[admin@Hautespot] > /port print
  # NAME                              USED-BY                           BAUD-
RATE
  0 serial0                           Serial Console                    9600
  1 databooster1                                                        9600
  2 databooster2                                                        9600
  3 databooster3                                                        9600
  4 databooster4                                                        9600
  5 databooster5                                                        9600
  6 databooster6                                                        9600
  7 databooster7                                                        9600
  8 databooster8                                                        9600
  9 cycladesA1                                                          9600
 10 cycladesA2                                                          9600
 11 cycladesA3                                                          9600
 12 cycladesA4                                                          9600
 13 cycladesA5                                                          9600
 14 cycladesA6                                                          9600
 15 cycladesA7                                                          9600
 16 cycladesA8                                                          9600
[admin@Hautespot] > set 9 baud-rate=38400
[admin@Hautespot] >
```

PPP Server Setup

Submenu level: ***/interface ppp-server***

Description

PPP server provides a remode connection service for users. When dialing in, the users can be authenticated locally using the local user database in the **/user** menu, or at the RADIUS server specified in the **/ip ppp** settings.

Property Description

port (*name*; default: **(unknown)**) - serial port
authentication (*multiple choice:* mschap2, mschap1, chap, pap; default: **mschap2, mschap1, chap, pap**) - authentication protocol
profile (*name*; default: **default**) - profile name used for the link
mtu (*integer*; default: **1500**) - Maximum Transmission Unit. Maximum packet size to be transmitted
mru (*integer*; default: **1500**) - Maximum Receive Unit
null-modem (no | yes; default: **no**) - enable/disable null-modem mode (when enabled, no modem initialization strings are sent)
modem-init (*text*; default: **""**) - modem initialization string. You may use **"s11=40"** to improve dialing speed
ring-count (*integer*; default: **1**) - number of rings to wait before answering phone
name (*name*; default: **ppp-inN**) - interface name for reference

Example

You can add a PPP server using the **add** command:

```
[admin@Hautespot] interface ppp-server> add name=test port=serial1
[admin@Hautespot] interface ppp-server> print
Flags: X - disabled, R - running
  0 X  name="test" mtu=1500 mru=1500 port=serial1
       authentication=mschap2,chap,pap profile=default modem-init=""
       ring-count=1 null-modem=no

[admin@Hautespot] interface ppp-server> enable 0
[admin@Hautespot] interface ppp-server> monitor test
            status: "waiting for call..."

[admin@Hautespot] interface ppp-server>
```

PPP Client Setup

Submenu level: ***/interface ppp-client***

Description

The section describes PPP clients configuration routines.

Property Description

port (*name*; default: **(unknown)**) - serial port
user (*text*; default: **""**) - P2P user name on the remote server to use for dialout
password (*text*; default: **""**) - P2P user password on the remote server to use for dialout

profile (*name*; default: **default**) - local profile to use for dialout
allow (*multiple choice:* mschap2, mschap1, chap, pap; default: **mschap2, mschap1, chap, pap**) - the protocol to allow the client to use for authentication
phone (*integer*; default: **""**) - phone number for dialout
tone-dial (yes | no; default: **yes**) - defines whether use tone dial or pulse dial
mtu (*integer*; default: **1500**) - Maximum Transmission Unit. Maximum packet size to be transmitted
mru (*integer*; default: **1500**) - Maximum Receive Unit
null-modem (no | yes; default: **no**) - enable/disable null-modem mode (when enabled, no modem initialization strings are sent)
modem-init (*text*; default: **""**) - modem initialization strings. You may use **"s11=40"** to improve dialing speed
dial-on-demand (yes | no; default: **no**) - enable/disable dial on demand
add-default-route (yes | no; default: **no**) - add PPP remote address as a default route
use-peer-dns (yes | no; default: **no**) - use DNS server settings from the remote server

Notes

Additional client profiles must be configured on the server side for clients to accomplish logon procedure. For more information see **Related Documents** section.

PPP client profiles must match at least partially (**local-address** and values related to encryption should match) with corresponding remote server values.

Example

You can add a PPP client using the **add** command:

```
[admin@Hautespot] interface ppp-client> add name=test user=test port=serial1 \
\... add-default-route=yes
[admin@Hautespot] interface ppp-client> print
Flags: X - disabled, R - running
  0 X  name="test" mtu=1500 mru=1500 port=serial1 user="test" password=""
       profile=default phone="" tone-dial=yes modem-init="" null-modem=no
       dial-on-demand=no add-default-route=yes use-peer-dns=no

[admin@Hautespot] interface ppp-client> enable 0
[admin@Hautespot] interface ppp-client> monitor test
[admin@Hautespot] interface ppp-client> monitor 0
           status: "dialing out..."

[admin@Hautespot] interface ppp-client>
```

PPP Application Example

Client - Server Setup

In this example we will consider the following network setup:

For a typical server setup we need to add one user to the **R1** and configure the PPP server.

```
[admin@Hautespot] ppp secret> add name=test password=test local-
address=3.3.3.1 \
\... remote-address=3.3.3.2
[admin@Hautespot] ppp secret> print
Flags: X - disabled
  0   name="test" service=any caller-id="" password="test" profile=default
      local-address=3.3.3.1 remote-address=3.3.3.2 routes=""

[admin@Hautespot] ppp secret> /int ppp-server
[admin@Hautespot] interface ppp-server> add port=serial1 disabled=no
[admin@Hautespot] interface ppp-server> print
Flags: X - disabled, R - running
  0    name="ppp-in1" mtu=1500 mru=1500 port=serial1
       authentication=mschap2,mschap1,chap,pap profile=default modem-init=""
       ring-count=1 null-modem=no

[admin@Hautespot] interface ppp-server>
```

Now we need to setup the client to connect to the server:

```
[admin@Hautespot] interface ppp-client> add port=serial1 user=test
password=test \
\... phone=132
[admin@Hautespot] interface ppp-client> print
Flags: X - disabled, R - running
  0 X  name="ppp-out1" mtu=1500 mru=1500 port=serial1 user="test"
       password="test" profile=default phone="132" tone-dial=yes
       modem-init="" null-modem=no dial-on-demand=no add-default-route=no
       use-peer-dns=no

[admin@Hautespot] interface ppp-client> enable 0

After a short duration of time the routers will be able to ping each other:
[admin@Hautespot] interface ppp-client> /ping 3.3.3.1
3.3.3.1 64 byte ping: ttl=64 time=43 ms
3.3.3.1 64 byte ping: ttl=64 time=11 ms
3.3.3.1 64 byte ping: ttl=64 time=12 ms
3.3.3.1 64 byte ping: ttl=64 time=11 ms
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 11/19.2/43 ms
[admin@Hautespot] interface ppp-client>
```

HauteRouterOS PPP User AAA

Document revision: 2.4 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

This documents provides summary, configuration reference and examples on PPP user management. This includes asynchronous PPP, PPTP, PPPoE and ISDN users.

Specifications

Packages required: ***system***
License required: *Level1*
Submenu level: ***/ppp***

Description

The HauteRouterOS provides scalable Authentication, Authorization and Accounting (AAA) functionality.

Local authentication is performed using the User Database and the Profile Database. The actual configuration for the given user is composed using respective user record from the User Database, associated item from the Profile Database and the item in the Profile database which is set as default for a given service the user is authenticating to. Default profile settings from the Profile database have lowest priority while the user access record settings from the User Database have highest priority with the only exception being particular IP addresses take precedence over IP pools in the **local-address** and **remote-address** settings, which described later on.

Support for RADIUS authentication gives the ISP or network administrator the ability to manage PPP user access and accounting from one server throughout a large network. The HauteRouterOS has a RADIUS client which can authenticate for PPP, PPPoE, PPTP, L2TP and ISDN connections. The attributes received from RADIUS server override the ones set in the default profile, but if some parameters are not received they are taken from the respective default profile.

Local PPP User Profiles

Submenu level: ***/ppp profile***

Description

PPP profiles are used to define default values for user access records stored under **/ppp secret** submenu. Settings in **/ppp secret** User Database override corresponding **/ppp profile** settings except that single IP addresses always take precedence over IP pools when specified as **local-address** or **remote-address** parameters.

Property Description

change-tcp-mss (yes | no | default; default: **default**) - modifies connection MSS settings
yes - adjust connection MSS value
no - do not atjust connection MSS value
default - accept this setting from the peer
dns-server (*IP address*{1,2}) - IP address of the DNS server to supply to clients
idle-timeout (*time*) - specifies the amount of time after which the link will be terminated if there was no activity present. There is no timeout set by default
0s - no link timeout is set
incoming-filter (*name*) - firewall chain name for incoming packets. Specified chain gets control for each packet coming from the client. The **ppp** chain should be manually added and rules with **action=jump jump-target=ppp** should be added to other relevant chains in order for this feature to work. For more information look at the Examples section
local-address (*IP address* | *name*) - IP address or IP address pool name for PPP server
name (*name*) - PPP profile name
only-one (yes | no | default; default: **default**) - defines whether a user is allowed to have more then one connection at a time
yes - a user is not allowed to have more than one connection at a time
no - the user is allowed to have more than one connection at a time
default - accept this setting from the peer
outgoing-filter (*name*) - firewall chain name for outgoing packets. Specified chain gets control for each packet going to the client. The **ppp** chain should be manually added and rules with **action=jump jump-target=ppp** should be added to other relevant chains in order for this feature to work. For more information look at the Examples section
rate-limit (*text*; default: **""**) - rate limitation in form of **rx-rate[/tx-rate] [rx-burst-rate[/tx-burst-rate] [rx-burst-threshold[/tx-burst-threshold] [rx-burst-time[/tx-burst-time] [priority] [rx-rate-min[/tx-rate-min]]]]** from the point of view of the router (so "rx" is client upload, and "tx" is client download). All rates are measured in bits per second, unless followed by optional 'k' suffix (kilobits per second) or 'M' suffix (megabits per second). If tx-rate is not specified, rx-rate serves as tx-rate too. The same applies for tx-burst-rate, tx-burst-threshold and tx-burst-time. If both rx-burst-threshold and tx-burst-threshold are not specified (but burst-rate is specified), rx-rate and tx-rate are used as burst thresholds. If both rx-burst-time and tx-burst-time are not specified, 1s is used as default. Priority takes values 1..8, where 1 implies the highest priority, but 8 - the lowest. If rx-rate-min and tx-rate-min are not specified rx-rate and tx-rate values are used. The rx-rate-min and tx-rate-min values can not exceed rx-rate and tx-rate values.
remote-address (*IP address* | *name*) - IP address or IP address pool name for PPP clients

session-timeout (*time*) - maximum time the connection can stay up. By default no time limit is set
0s - no connection timeout
use-compression (yes | no | default; default: **default**) - specifies whether to use data compression or not
yes - enable data compression
no - disable data compression
default - accept this setting from the peer
use-encryption (yes | no | default; default: **default**) - specifies whether to use data encryption or not
yes - enable data encryption
no - disable data encryption
default - accept this setting from the peer
use-vj-compression (yes | no | default; default: **default**) - specifies whether to use Van Jacobson header compression algorithm
yes - enable Van Jacobson header compression
no - disable Van Jacobson header compression
default - accept this setting from the peer
wins-server (*IP address*{1,2}) - IP address of the WINS server to supply to Windows clients

Notes

There are two default profiles that cannot be removed:

```
[admin@rb13] ppp profile> print
Flags: * - default
 0 * name="default" use-compression=no use-vj-compression=no use-encryption=no
only-one=no
     change-tcp-mss=yes
 1 * name="default-encryption" use-compression=default use-vj-
compression=default use-encryption=yes
     only-one=default change-tcp-mss=default
[admin@rb13] ppp profile>
```

Use Van Jacobson compression only if you have to because it may slow down the communications on bad or congested channels.

incoming-filter and **outgoing-filter** arguments add dynamic **jump** rules to chain **ppp**, where the **jump-target** argument will be equal to **incoming-filter** or **outgoing-filter** argument in **/ppp profile**. Therefore, chain **ppp** should be manually added before changing these arguments.

only-one parameter is ignored if RADIUS authentication is used

Example

To add the profile **ex** that assigns the router itself the **10.0.0.1** address, and the addresses from the **ex** pool to the clients, filtering traffic coming from clients through **mypppclients** chain:

```
[admin@rb13] ppp profile> add name=ex local-address=10.0.0.1 remote-address=ex
incoming-filter=mypppclients
[admin@rb13] ppp profile> print
Flags: * - default
 0 * name="default" use-compression=no use-vj-compression=no use-encryption=no
only-one=no
     change-tcp-mss=yes
 1   name="ex" local-address=10.0.0.1 remote-address=ex use-
compression=default
     use-vj-compression=default use-encryption=default only-one=default
change-tcp-mss=default
     incoming-filter=mypppclients
 2 * name="default-encryption" use-compression=default use-vj-
compression=default use-encryption=yes
     only-one=default change-tcp-mss=default
[admin@rb13] ppp profile>
```

Local PPP User Database

Submenu level: ***/ppp secret***

Description

PPP User Database stores PPP user access records with PPP user profile assigned to each user.

Property Description

caller-id (*text*; default: **""**) - for **PPTP** and **L2TP** it is the IP address a client must connect from. For **PPPoE** it is the MAC address (written in CAPITAL letters) a client must connect from. For **ISDN** it is the caller's number (that may or may not be provided by the operator) the client may dial-in from
"" - no restrictions on where clients may connect from
limit-bytes-in (*integer*; default: **0**) - maximal amount a client can upload, in bytes, for a session
limit-bytes-out (*integer*; default: **0**) - maximal amount a client can download, in bytes, for a session
local-address (*IP address* | *name*) - IP address or IP address pool name for PPP server
name (*name*) - user's name used for authentication
password (*text*; default: **""**) - user's password used for authentication
profile (*name*; default: **default**) - profile name to use together with this access record for user authentication
remote-address (*IP address* | *name*) - IP address or IP address pool name for PPP clients
routes (*text*) - routes that appear on the server when the client is connected. The route format is: **dst-address gateway metric** (for example, **10.1.0.0/ 24 10.0.0.1 1**). Several routes may be specified separated with commas

service (any | async | isdn | l2tp | pppoe | pptp; default: **any**) - specifies the services available to a particular user

Example

To add the user **ex** with password **lkjrht** and profile **ex** available for PPTP service only, enter the following command:

```
[admin@rb13] ppp secret> add name=ex password=lkjrht service=pptp profile=ex
[admin@rb13] ppp secret> print
Flags: X - disabled
 #   NAME                SERVICE CALLER-ID         PASSWORD          PROFILE
REMOTE-ADDRESS
 0   ex                  pptp                      lkjrht            ex
0.0.0.0
[admin@rb13] ppp secret>
```

Monitoring Active PPP Users

Command name: ***/ppp active print***

Property Description

address (*read-only: IP address*) - IP address the client got from the server
bytes (*read-only: integer/integer*) - amount of bytes transfered through tis connection. First figure represents amount of transmitted traffic from the router's point of view, while the second one shows amount of received traffic
caller-id (*read-only: text*) - for **PPTP** and **L2TP** it is the IP address the client connected from. For **PPPoE** it is the MAC address the client connected from. For **ISDN** it is the caller's number the client dialed-in from
"" - no restrictions on where clients may connect from
encoding (*read-only: text*) - shows encryption and encoding (separated with '/' if asymmetric) being used in this connection
limit-bytes-in (*read-only: integer*) - maximal amount of bytes the user is allowed to send to the router
limit-bytes-out (*read-only: integer*) - maximal amount of bytes the router is allowed to send to the client
name (*read-only: name*) - user name supplied at authentication stage
packets (*read-only: integer/integer*) - amount of packets transfered through tis connection. First figure represents amount of transmitted traffic from the router's point of view, while the second one shows amount of received traffic
service (*read-only:* async | isdn | l2tp | pppoe | pptp) - the type of service the user is using
session-id (*read-only: text*) - shows unique client identifier
uptime (*read-only: time*) - user's uptime

Example

```
[admin@rb13] > /ppp active print
Flags: R - radius
 #   NAME          SERVICE CALLER-ID         ADDRESS         UPTIME   ENCODING
 0   ex            pptp    10.0.11.12        10.0.0.254      1m16s
MPPE128...
[admin@rb13] > /ppp active print detail
Flags: R - radius
 0   name="ex" service=pptp caller-id="10.0.11.12" address=10.0.0.254
     uptime=1m22s encoding="MPPE128 stateless" session-id=0x8180002B
     limit-bytes-in=200000000 limit-bytes-out=0
[admin@rb13] > /ppp active print stats
Flags: R - radius
 #   NAME          BYTES                 PACKETS
 0   ex            10510/159690614       187/210257
[admin@rb13] >
```

PPP User Remote AAA

Submenu level: ***/ppp aaa***

Property Description

accounting (yes | no; default: **yes**) - enable RADIUS accounting
interim-update (*time*; default: **0s**) - Interim-Update time interval
use-radius (yes | no; default: **no**) - enable user authentication via RADIUS

Notes

RADIUS user database is consulted only if the required username is not found in local user database.

Example

To enable RADIUS AAA:

```
[admin@Hautespot] ppp aaa> set use-radius=yes
[admin@Hautespot] ppp aaa> print
        use-radius: yes
        accounting: yes
    interim-update: 0s
[admin@Hautespot] ppp aaa>
```

HauteRouterOS PPPoE

Document revision: 1.5 (Tuesday, October 30, 2007)

Applies to: HauteRouterOS V3.0

General Information

Summary

The PPPoE (Point to Point Protocol over Ethernet) protocol provides extensive user management, network management and accounting benefits to ISPs and network administrators. Currently PPPoE is used mainly by ISPs to control client connections for xDSL and cable modems as well as plain Ethernet networks. PPPoE is an extension of the standard Point to Point Protocol (PPP). The difference between them is expressed in transport method: PPPoE employs Ethernet instead of modem connection.

Generally speaking, PPPoE is used to hand out IP addresses to clients based on the user (and workstation, if desired) authentication as opposed to workstation only authentication, when static IP addresses or DHCP are used. It is adviced not to use static IP addresses or DHCP on the same interfaces as PPPoE for obvious security reasons.

HauteRouterOS can act as a RADIUS client - you can use a RADIUS server to authenticate PPPoE clients and use accounting for them.

A PPPoE connection is composed of a client and an access concentrator (server). The client may be any computer that has the PPPoE client protocol support installed. The HauteRouterOS supports both - client and access concentrator implementations of PPPoE. The PPPoE client and server work over any Ethernet level interface on the router - wireless 802.11 (Aironet, Cisco, WaveLan, Prism, Atheros), 10/100/1000 Mbit/s Ethernet, RadioLan and EoIP (Ethernet over IP tunnel). No encryption, MPPE 40bit RSA and MPPE 128bit RSA encryption is supported.

Note that when RADIUS server is authenticating a user with CHAP, MS-CHAPv1 or MS-CHAPv2, the RADIUS protocol does not use shared secret, it is used only in authentication reply. So if you have a wrong shared secret, RADIUS server will accept the request. You can use **/radius monitor** command to see **bad-replies** parameter. This value should increase whenever a client tries to connect.

Supported connections

- HauteRouterOS PPPoE client to any PPPoE server (access concentrator)
- HauteRouterOS server (access concentrator) to multiple PPPoE clients (clients are avaliable for almost all operating systems and most routers)

Quick Setup Guide

- To configure HauteRouterOS to be a PPPoE client
    1. Just add a pppoe-client:

```
2. /interface pppoe-client add name=pppoe-user-mike user=mike
   password=123 \
```

```
\... interface=wlan1 service-name=internet disabled=no
```

- To configure HauteRouterOS to be an Access Concentrator (PPPoE Server)
  1. Add an address pool for the clients from **10.1.1.62** to **10.1.1.72**, called pppoe-pool:

```
/ip pool add name="pppoe-pool" ranges=10.1.1.62-10.1.1.72
```

  2. Add PPP profile, called **pppoe-profile** where **local-address** will be the router's address and clients will have an address from **pppoe-pool**:

```
/ppp profile add name="pppoe-profile" local-address=10.1.1.1
remote-address=pppoe-pool
```

  3. Add a user with username **mike** and password **123**:

```
/ppp secret add name=mike password=123 service=pppoe
profile=pppoe-profile
```

  4. Now add a pppoe server:

```
5. /interface pppoe-server server add service-name=internet
   interface=wlan1 \
```

```
\... default-profile=pppoe-profile
```

Specifications

Packages required: ***ppp***
License required: *Level1 (limited to 1 interface) , Level3 (limited to 200 interfaces) , Level4 (limited to 200 interfaces) , Level5 (limited to 500 interfaces) , Level6 (unlimited)*
Submenu level: ***/interface pppoe-server***, ***/interface pppoe-client***
Standards and Technologies: PPPoE (RFC 2516)
Hardware usage: *PPPoE server may require additional RAM (uses approx. 9KiB (plus extra 10KiB for packet queue, if data rate limitation is used) for each connection) and CPU power. Maximum of 65535 connections is supported.*

Additional Resources

Links for PPPoE documentation:

- http://www.faqs.org/rfcs/rfc2516.html

PPPoE Clients:

- RASPPPoE for Windows 95, 98, 98SE, ME, NT4, 2000, XP, .NET

  http://www.raspppoe.com/

PPPoE Client Setup

Submenu level: ***/interface pppoe-client***

Description

The PPPoE client supports high-speed connections. It is fully compatible with the Hautespot PPPoE server (access concentrator).

Note for Windows. Some connection instructions may use the form where the "phone number", such as "Hautespot_AC\mt1", to indicate that "Hautespot_AC" is the access concentrator name and "mt1" is the service name.

Property Description

ac-name (*text*; default: **""**) - this may be left blank and the client will connect to any access concentrator that offers the "service" name selected
add-default-route (yes | no; default: **no**) - whether to add a default route automatically
allow (*multiple choice:* mschap2, mschap1, chap, pap; default: **mschap2, mschap1, chap, pap**) - the protocol to allow the client to use for authentication
dial-on-demand (yes | no; default: **no**) - connects to AC only when outbound traffic is generated and disconnects when there is no traffic for the period set in the idle-timeout value
interface (*name*) - interface the PPPoE server can be connected through
mru (*integer*; default: **1480**) - Maximum Receive Unit. The optimal value is the MTU of the interface the tunnel is working over decreased by 20 (so, for 1500-byte ethernet link, set the MTU to 1480 to avoid fragmentation of packets)
mtu (*integer*; default: **1480**) - Maximum Transmission Unit. The optimal value is the MTU of the interface the tunnel is working over decreased by 20 (so, for 1500-byte ethernet link, set the MTU to 1480 to avoid fragmentation of packets)
name (*name*; default: **pppoe-out1**) - name of the PPPoE interface
password (*text*; default: **""**) - a user password used to connect the PPPoE server
profile (*name*) - default profile for the connection
service-name (*text*; default: **""**) - specifies the service name set on the access concentrator. Leave it blank unless you have many services and need to specify the one you need to connect to

use-peer-dns (yes | no; default: **no**) - whether to set the router's default DNS to the PPP peer DNS (i.e. whether to get DNS settings from the peer)
user (*text*; default: **""**) - a user name that is present on the PPPoE server

Example

To add and enable PPPoE client on the **gig** interface connecting to the AC that provides **testSN** service using user name **john** with the password **password**:

```
[admin@RemoteOffice] interface pppoe-client> add interface=gig \
\... service-name=testSN user=john password=password disabled=no
[admin@RemoteOffice] interface pppoe-client> print
Flags: X - disabled, R - running
  0  R name="pppoe-out1" mtu=1480 mru=1480 interface=gig user="john"
       password="password" profile=default service-name="testSN" ac-name=""
       add-default-route=no dial-on-demand=no use-peer-dns=no
```

Monitoring PPPoE Client

Command name: ***/interface pppoe-client monitor***

Property Description

ac-mac (*MAC address*) - MAC address of the access concentrator (AC) the client is connected to
ac-name (*text*) - name of the AC the client is connected to
encoding (*text*) - encryption and encoding (if asymmetric, separated with '/') being used in this connection
service-name (*text*) - name of the service the client is connected to
status (*text*) - status of the client
Dialing - attempting to make a connection
Verifying password... - connection has been established to the server, password verification in progress
Connected - self-explanatory
Terminated - interface is not enabled or the other side will not establish a connection **uptime** (time) - connection time displayed in days, hours, minutes and seconds
uptime (*time*) - connection time displayed in days, hours, minutes and seconds

Example

To monitor the **pppoe-out1** connection:

```
[admin@Hautespot] interface pppoe-client> monitor pppoe-out1
          status: "connected"
          uptime: 10s
        encoding: "none"
    service-name: "testSN"
         ac-name: "10.0.0.1"
```

```
          ac-mac: 00:C0:DF:07:5E:E6

[admin@Hautespot] interface pppoe-client>
```

PPPoE Server Setup (Access Concentrator)

Submenu level: ***/interface pppoe-server server***

Description

The PPPoE server (access concentrator) supports multiple servers for each interface - with differing service names. Currently the throughput of the PPPoE server has been tested to 160 Mb/s on a Celeron 600 CPU. Using higher speed CPUs, throughput should increase proportionately.

The **access concentrator name** and PPPoE **service name** are used by clients to identity the access concentrator to register with. The **access concentrator name** is the same as the **identity** of the router displayed before the command prompt. The identity may be set within the **/system identity** submenu.

PPPoE users are created in **/ppp secret** menu, see the AAA manual for further information.

Note that if no service name is specified in WindowsXP, it will use only service with no name. So if you want to serve WindowsXP clients, leave your service name empty.

Property Description

authentication (*multiple choice:* mschap2 | mschap1 | chap | pap; default: **mschap2, mschap1, chap, pap**) - authentication algorithm
default-profile (*name*; default: **default**) - default profile to use
interface (*name*) - interface to which the clients will connect to
keepalive-timeout (*time*; default: **10**) - defines the time period (in seconds) after which the router is starting to send keepalive packets every second. If no traffic and no keepalive responses has came for that period of time (i.e. 2 * keepalive-timeout), not responding client is proclaimed disconnected.
max-mru (*integer*; default: **1480**) - Maximum Receive Unit. The optimal value is the MTU of the interface the tunnel is working over decreased by 20 (so, for 1500-byte Ethernet link, set the MTU to 1480 to avoid fragmentation of packets)
max-mtu (*integer*; default: **1480**) - Maximum Transmission Unit. The optimal value is the MTU of the interface the tunnel is working over decreased by 20 (so, for 1500-byte Ethernet link, set the MTU to 1480 to avoid fragmentation of packets)
max-sessions (*integer*; default: **0**) - maximum number of clients that the AC can serve

0 - unlimited
one-session-per-host (yes | no; default: **no**) - allow only one session per host (determined by MAC address). If a host will try to establish a new session, the old one will be closed
service-name (*text*) - the PPPoE service name

Notes

The default **keepalive-timeout** value of **10** is OK in most cases. If you set it to **0**, the router will not disconnect clients until they log out or router is restarted. To resolve this problem, the **one-session-per-host** property can be used.

Security issue: do not assign an IP address to the interface you will be receiving the PPPoE requests on.

Example

To add PPPoE server on **ether1** interface providing **ex** service and allowing only one connection per host:

```
[admin@Hautespot] interface pppoe-server server> add interface=ether1 \
\... service-name=ex one-session-per-host=yes
[admin@Hautespot] interface pppoe-server server> print
Flags: X - disabled
  0 X service-name="ex" interface=ether1 mtu=1480 mru=1480
      authentication=mschap2,mschap,chap,pap keepalive-timeout=10
      one-session-per-host=yes default-profile=default

[admin@Hautespot] interface pppoe-server server>
```

PPPoE Server Users

Submenu level: ***/interface pppoe-server***

Property Description

encoding (*read-only: text*) - encryption and encoding (if asymmetric, separated with '/') being used in this connection
name (*name*) - interface name
remote-address (*read-only: MAC address*) - MAC address of the connected client
service-name (*name*) - name of the service the user is connected to
uptime (*time*) - shows how long the client is connected
user (*name*) - the name of the connected user

Example

To view the currently connected users:

```
[admin@Hautespot] interface pppoe-server> print
Flags: R - running
  #    NAME        SERVICE REMOTE-ADDRESS    USER     ENCO... UPTIME
  0 R <pppoe-ex> ex      00:C0:CA:16:16:A5 ex              12s

[admin@Hautespot] interface pppoe-server>
```

To disconnect the user **ex**:

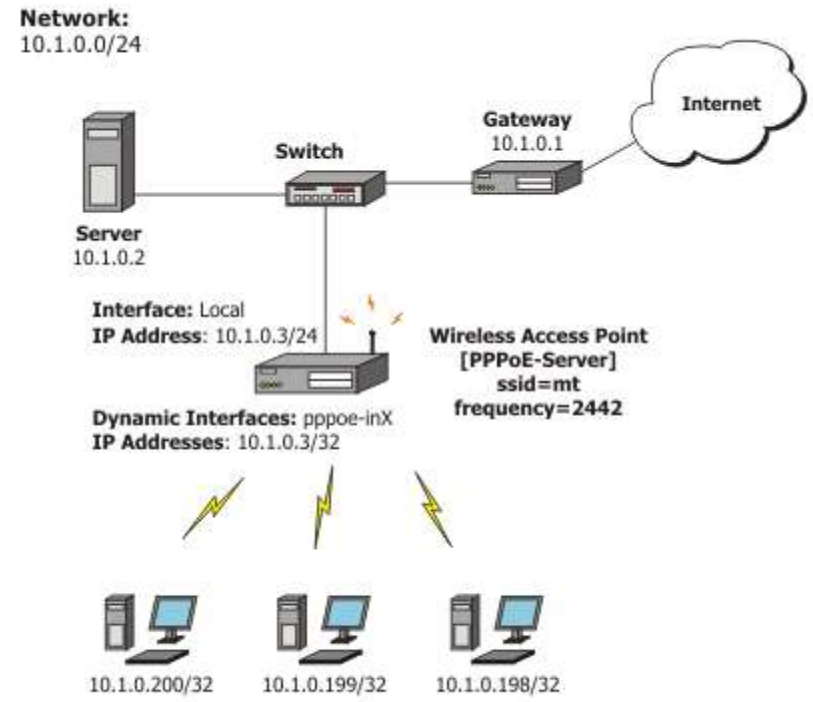
```
[admin@Hautespot] interface pppoe-server> remove [find user=ex]
[admin@Hautespot] interface pppoe-server> print

[admin@Hautespot] interface pppoe-server>
```

Application Examples

PPPoE in a multipoint wireless 802.11g network

In a wireless network, the PPPoE server may be attached to an Access Point (as well as to a regular station of wireless infrastructure). Either our RouterOS client or Windows PPPoE clients may connect to the Access Point for PPPoE authentication. Further, for RouterOS clients, the radio interface may be set to MTU 1600 so that the PPPoE interface may be set to MTU 1500. This optimizes the transmission of 1500 byte packets and avoids any problems associated with MTUs lower than 1500. It has not been determined how to change the MTU of the Windows wireless interface at this moment.

Let us consider the following setup where the Hautespot Wireless AP offers wireless clients transparent access to the local network with authentication:



First of all, the wireless interface should be configured:

```
[admin@PPPoE-Server] interface wireless> set 0 mode=ap-bridge \
   frequency=2442 band=2.4ghz-b/g ssid=mt disabled=no
[admin@PPPoE-Server] interface wireless> print
Flags: X - disabled, R - running
 0    name="wlan1" mtu=1500 mac-address=00:01:24:70:53:04 arp=enabled
      disable-running-check=no interface-type=Atheros AR5211
      radio-name="000124705304" mode=station ssid="mt" area=""
      frequency-mode=superchannel country=no_country_set antenna-gain=0
      frequency=2412 band=2.4ghz-b scan-list=default rate-set=default
      supported-rates-b=1Mbps,2Mbps,5.5Mbps,11Mbps
      supported-rates-a/g=6Mbps,9Mbps,12Mbps,18Mbps,24Mbps,36Mbps,48Mbps,
                          54Mbps
      basic-rates-b=1Mbps basic-rates-a/g=6Mbps max-station-count=2007
      ack-timeout=dynamic tx-power=default tx-power-mode=default
      noise-floor-threshold=default periodic-calibration=default
      burst-time=disabled fast-frames=no dfs-mode=none antenna-mode=ant-a
      wds-mode=disabled wds-default-bridge=none wds-ignore-ssid=no
      update-stats-interval=disabled default-authentication=yes
      default-forwarding=yes default-ap-tx-limit=0 default-client-tx-limit=0
      hide-ssid=no security-profile=default disconnect-timeout=3s
      on-fail-retry-time=100ms preamble-mode=both
[admin@PPPoE-Server] interface wireless>
```

Now, configure the Ethernet interface, add the IP address and set the default route:

```
[admin@PPPoE-Server] ip address> add address=10.1.0.3/24 interface=Local
[admin@PPPoE-Server] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
 #   ADDRESS            NETWORK         BROADCAST       INTERFACE
 0   10.1.0.3/24        10.1.0.0        10.1.0.255      Local
[admin@PPPoE-Server] ip address> /ip route
[admin@PPPoE-Server] ip route> add gateway=10.1.0.1
[admin@PPPoE-Server] ip route> print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf
 #     DST-ADDRESS        G GATEWAY         DISTANCE INTERFACE
 0 ADC 10.1.0.0/24                                   Local
 1 A S 0.0.0.0/0          r 10.1.0.1        1        Local
[admin@PPPoE-Server] ip route> /interface ethernet
[admin@PPPoE-Server] interface ethernet> set Local arp=proxy-arp
[admin@PPPoE-Server] interface ethernet> print
Flags: X - disabled, R - running
 #    NAME                                    MTU   MAC-ADDRESS       ARP
 0  R Local                                   1500  00:0C:42:03:25:53 proxy-arp
[admin@PPPoE-Server] interface ethernet>
```

We should add PPPoE server to the wireless interface:

```
[admin@PPPoE-Server] interface pppoe-server server> add interface=wlan1 \
   service-name=mt one-session-per-host=yes disabled=no
[admin@PPPoE-Server] interface pppoe-server server> print
Flags: X - disabled
 0   service-name="mt" interface=wlan1 max-mtu=1480 max-mru=1480
     authentication=pap,chap,mschap1,mschap2 keepalive-timeout=10
     one-session-per-host=yes max-sessions=0 default-profile=default
[admin@PPPoE-Server] interface pppoe-server server>
```

Finally, we can set up PPPoE clients:

```
[admin@PPPoE-Server] ip pool> add name=pppoe ranges=10.1.0.100-10.1.0.200
[admin@PPPoE-Server] ip pool> print
 # NAME                                      RANGES
 0 pppoe                                     10.1.0.100-10.1.0.200
[admin@PPPoE-Server] ip pool> /ppp profile
[admin@PPPoE-Server] ppp profile> set default use-encryption=yes \
   local-address=10.1.0.3 remote-address=pppoe
[admin@PPPoE-Server] ppp profile> print
Flags: * - default
 0 * name="default" local-address=10.1.0.3 remote-address=pppoe
     use-compression=no use-vj-compression=no use-encryption=yes only-one=no
     change-tcp-mss=yes

 1 * name="default-encryption" use-compression=default
     use-vj-compression=default use-encryption=yes only-one=default
     change-tcp-mss=default
[admin@PPPoE-Server] ppp profile> .. secret
[admin@PPPoE-Server] ppp secret> add name=w password=wkst service=pppoe
[admin@PPPoE-Server] ppp secret> add name=l password=ltp service=pppoe
[admin@PPPoE-Server] ppp secret> print
```

```
Flags: X - disabled
 #   NAME         SERVICE CALLER-ID PASSWORD  PROFILE            REMOTE-ADDRESS
 0   w            pppoe             wkst      default            0.0.0.0
 1   l            pppoe             ltp       default            0.0.0.0
[admin@PPPoE-Server] ppp secret>
```

Thus we have completed the configuration and added two users: **w** and **l** who are able to connect to Internet, using PPPoE client software.

Note that Windows XP built-in client supports encryption, but RASPPPOE does not. So, if it is planned not to support Windows clients older than Windows XP, it is recommended to switch **require-encryption** to **yes** value in the **default** profile configuration. In other case, the server will accept clients that do not encrypt data.

Troubleshooting

Description

- **I can connect to my PPPoE server. The ping goes even through it, but I still cannot open web pages**

  Make sure that you have specified a valid DNS server in the router (in **/ip dns** or in **/ppp profile** the **dns-server** parameter).

- **The PPPoE server shows more than one active user entry for one client, when the clients disconnect, they are still shown and active**

  Set the **keepalive-timeout** parameter (in the PPPoE server configuration) to **10** if You want clients to be considered logged off if they do not respond for 10 seconds.

  Note that if the **keepalive-timeout** parameter is set to **0** and the **only-one** parameter (in PPP profile settings) is set to **yes** then the clients might be able to connect only once. To resolve this problem **one-session-per-host** parameter in PPPoE server configuration should be set to **yes**

- **I can get through the PPPoE link only small packets (eg. pings)**

  You need to change **mss** of all the packets passing through the PPPoE link to the value of PPPoE link's MTU-40 at least on one of the peers. So for PPPoE link with MTU of 1480:

```
[admin@MT] interface pppoe-server server> set 0 max-mtu=1440 max-
mru=1440
[admin@MT] interface pppoe-server server> print
Flags: X - disabled
 0   service-name="mt" interface=wlan1 max-mtu=1440 max-mru=1440
```

```
      authentication=pap,chap,mschap1,mschap2 keepalive-timeout=10
      one-session-per-host=yes max-sessions=0 default-profile=default
[admin@MT] interface pppoe-server server>
```

- **My windows PPPoE client obtains IP address and default gateway from the Hautespot PPPoE server, but it cannot ping beyond the PPPoE server and use the Internet**

  PPPoE server is not bridging the clients. Configure masquerading for the PPPoE client addresses, or make sure you have proper routing for the address space used by the clients, or you enable Proxy-ARP on the Ethernet interface (See the IP Addresses and Address Resolution Protocol (ARP) Manual)

- **My Windows XP client cannot connect to the PPPoE server**

  You have to specify the "Service Name" in the properties of the XP PPPoE client. If the service name is not set, or it does not match the service name of the Hautespot PPPoE server, you get the "line is busy" errors, or the system shows "verifying password - unknown error"

- **I want to have logs for PPPoE connection establishment**

  Configure the logging feature under the **/system logging facility** and enable the PPP type logs

HauteRouterOS PPTP

Document revision: 1.4 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

PPTP (Point to Point Tunnel Protocol) supports encrypted tunnels over IP. The HauteRouterOS implementation includes support for PPTP client and server.

General applications of PPTP tunnels:

- For secure router-to-router tunnels over the Internet
- To link (bridge) local Intranets or LANs (when EoIP is also used)

- For mobile or remote clients to remotely access an Intranet/LAN of a company (see PPTP setup for Windows for more information)

Each PPTP connection is composed of a server and a client. The HauteRouterOS may function as a server or client - or, for various configurations, it may be the server for some connections and client for other connections. For example, the client created below could connect to a Windows 2000 server, another Hautespot Router, or another router which supports a PPTP server.

Quick Setup Guide

To make a PPTP tunnel between 2 Hautespot routers with IP addresses **10.5.8.104** (PPTP server) and **10.1.0.172** (PPTP client), follow the next steps.

- Setup on PPTP server:
  1. Add a user:

```
2. [admin@PPTP-Server] ppp secret> add name=jack password=pass \
```

```
\... local-address=10.0.0.1 remote-address=10.0.0.2
```

  3. Enable the PPTP server:

```
[admin@PPTP-Server] interface pptp-server server> set enabled=yes
```

- Setup on PPTP client:
  1. Add the PPTP client:

```
2. [admin@PPTP-Client] interface pptp-client> add user=jack
   password=pass \
```

```
\... connect-to=10.5.8.104 disabled=no
```

Specifications

Packages required: ***ppp***
License required: *Level1 (limited to 1 tunnel)* , *Level3 (limited to 200 tunnels)* , *Level5*
Submenu level: ***/interface pptp-server***, ***/interface pptp-client***
Standards and Technologies: PPTP (RFC 2637)
Hardware usage: *Not significant*

Description

PPTP is a secure tunnel for transporting IP traffic using PPP. PPTP encapsulates PPP in virtual lines that run over IP. PPTP incorporates PPP and MPPE (Microsoft Point to

Point Encryption) to make encrypted links. The purpose of this protocol is to make well-managed secure connections between routers as well as between routers and PPTP clients (clients are available for and/or included in almost all OSs including Windows).

PPTP includes PPP authentication and accounting for each PPTP connection. Full authentication and accounting of each connection may be done through a RADIUS client or locally.

MPPE 40bit RC4 and MPPE 128bit RC4 encryption are supported.

PPTP traffic uses TCP port 1723 and IP protocol GRE (Generic Routing Encapsulation, IP protocol ID 47), as assigned by the Internet Assigned Numbers Authority (IANA). PPTP can be used with most firewalls and routers by enabling traffic destined for TCP port 1723 and protocol 47 traffic to be routed through the firewall or router.

PPTP connections may be limited or impossible to setup though a masqueraded/NAT IP connection. Please see the Microsoft and RFC links at the end of this section for more information.

Additional Resources

- http://msdn.microsoft.com/library/backgrnd/html/understanding_pptp.htm
- http://support.microsoft.com/support/kb/articles/q162/8/47.asp
- http://www.ietf.org/rfc/rfc2637.txt?number=2637
- http://www.ietf.org/rfc/rfc3078.txt?number=3078
- http://www.ietf.org/rfc/rfc3079.txt?number=3079

PPTP Client Setup

Submenu level: ***/interface pptp-client***

Property Description

add-default-route (yes | no; default: **no**) - whether to use the server which this client is connected to as its default router (gateway)
allow (*multiple choice:* mschap2, mschap1, chap, pap; default: **mschap2, mschap1, chap, pap**) - the protocol to allow the client to use for authentication
connect-to (*IP address*) - The IP address of the PPTP server to connect to
mru (*integer*; default: **1460**) - Maximum Receive Unit. The optimal value is the MRU of the interface the tunnel is working over decreased by 40 (so, for 1500-byte ethernet link, set the MRU to 1460 to avoid fragmentation of packets)
mtu (*integer*; default: **1460**) - Maximum Transmission Unit. The optimal value is the MTU of the interface the tunnel is working over decreased by 40 (so, for 1500-byte ethernet link, set the MTU to 1460 to avoid fragmentation of packets)
name (*name*; default: **pptp-outN**) - interface name for reference
password (*text*; default: **""**) - user password to use when logging to the remote server

profile (*name*; default: **default**) - profile to use when connecting to the remote server
user (*text*) - user name to use when logging on to the remote server

Example

To set up PPTP client named **test2** using unsername **john** with password **john** to connect to the **10.1.1.12** PPTP server and use it as the default gateway:

```
[admin@Hautespot] interface pptp-client> add name=test2 connect-to=10.1.1.12 \
\... user=john add-default-route=yes password=john
[admin@Hautespot] interface pptp-client> print
Flags: X - disabled, R - running
  0 X  name="test2" mtu=1460 mru=1460 connect-to=10.1.1.12 user="john"
       password="john" profile=default add-default-route=yes


[admin@Hautespot] interface pptp-client> enable 0
```

Monitoring PPTP Client

Command name: ***/interface pptp-client monitor***

Property Description

encoding (*text*) - encryption and encoding (if asymmetric, seperated with '/') being used in this connection
status (*text*) - status of the client
Dialing - attempting to make a connection
Verifying password... - connection has been established to the server, password verification in progress
Connected - self-explanatory
Terminated - interface is not enabled or the other side will not establish a connection **uptime** (time) - connection time displayed in days, hours, minutes and seconds
uptime (*time*) - connection time displayed in days, hours, minutes and seconds

Example

Example of an established connection:

```
[admin@Hautespot] interface pptp-client> monitor test2
      uptime: 4h35s
    encoding: MPPE 128 bit, stateless
      status: Connected
[admin@Hautespot] interface pptp-client>
```

PPTP Server Setup

Submenu level: ***/interface pptp-server server***

Description

The PPTP server creates a dynamic interface for each connected PPTP client. The PPTP connection count from clients depends on the license level you have. Level1 license allows 1 PPTP client, Level3 or Level4 licenses up to 200 clients, and Level5 or Level6 licenses do not have PPTP client limitations.

To create PPTP users, you should consult the PPP secret and PPP Profile manuals. It is also possible to use the Hautespot router as a RADIUS client to register the PPTP users, see the manual how to do it.

Property Description

authentication (*multiple choice:* pap | chap | mschap1 | mschap2; default: **mschap2**) - authentication algorithm
default-profile - default profile to use
enabled (yes | no; default: **no**) - defines whether PPTP server is enabled or not
keepalive-timeout (*time*; default: **30**) - defines the time period (in seconds) after which the router is starting to send keepalive packets every second. If no traffic and no keepalive responses has came for that period of time (i.e. 2 * keepalive-timeout), not responding client is proclaimed disconnected
mru (*integer*; default: **1460**) - Maximum Receive Unit. The optimal value is the MRU of the interface the tunnel is working over decreased by 40 (so, for 1500-byte ethernet link, set the MRU to 1460 to avoid fragmentation of packets)
mtu (*integer*; default: **1460**) - Maximum Transmission Unit. The optimal value is the MTU of the interface the tunnel is working over decreased by 40 (so, for 1500-byte ethernet link, set the MTU to 1460 to avoid fragmentation of packets)

Example

To enable PPTP server:

```
[admin@Hautespot] interface pptp-server server> set enabled=yes
[admin@Hautespot] interface pptp-server server> print
               enabled: yes
                   mtu: 1460
                   mru: 1460
        authentication: mschap2,mschap1
    keepalive-timeout: 30
      default-profile: default
[admin@Hautespot] interface pptp-server server>
```

PPTP Server Users

Submenu level: ***/interface pptp-server***

Description

There are two types of items in PPTP server configuration - static users and dynamic connections. A dynamic connection can be established if the user database or the **default-profile** has its **local-address** and **remote-address** set correctly. When static users are added, the default profile may be left with its default values and only PPP user (in **/ppp secret**) should be configured. **Note** that in both cases PPP users must be configured properly.

Property Description

client-address (*IP address*) - shows (cannot be set here) the IP address of the connected client
encoding (*text*) - encryption and encoding (if asymmetric, separated with '/') being used in this connection
mtu (*integer*) - (cannot be set here) client's MTU
name (*name*) - interface name
uptime (*time*) - shows how long the client is connected
user (*name*) - the name of the user that is configured statically or added dynamically

Example

To add a static entry for **ex1** user:

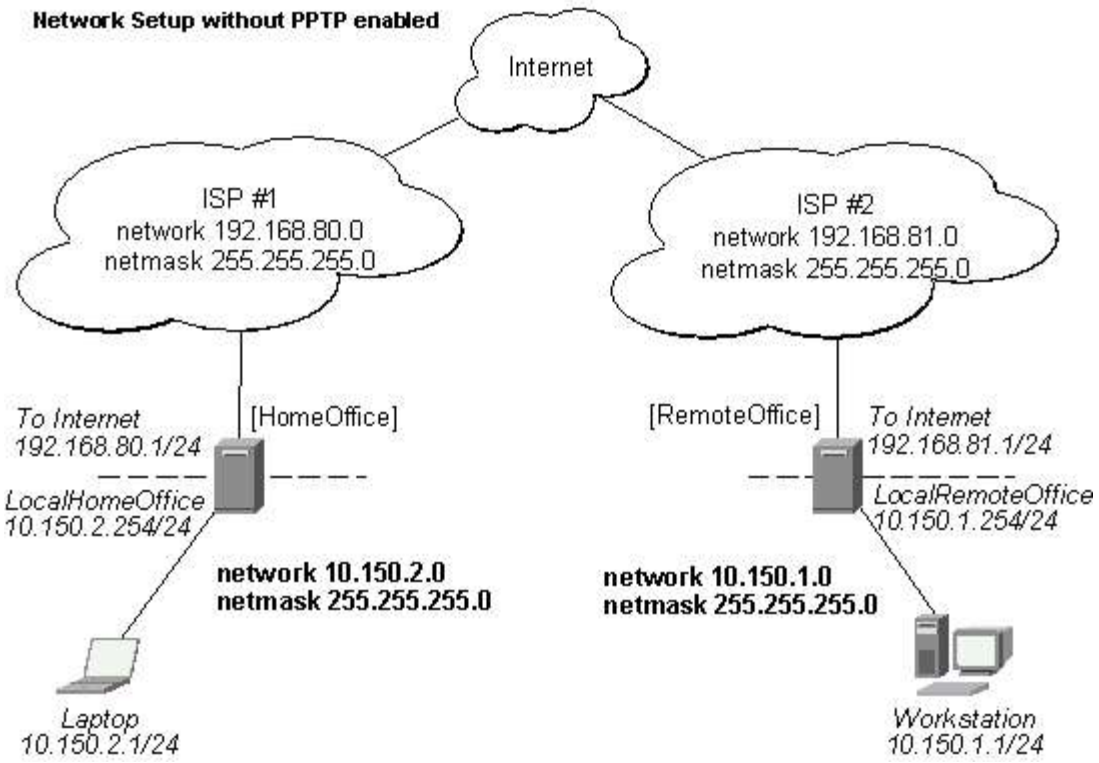
```
[admin@Hautespot] interface pptp-server> add user=ex1
[admin@Hautespot] interface pptp-server> print
Flags: X - disabled, D - dynamic, R - running
  #     NAME                 USER         MTU   CLIENT-ADDRESS  UPTIME
ENC...
  0  DR <pptp-ex>            ex           1460  10.0.0.202      6m32s    none
  1     pptp-in1             ex1
[admin@Hautespot] interface pptp-server>
```

In this example an already connected user **ex** is shown besides the one we just added.

PPTP Application Examples

Router-to-Router Secure Tunnel Example

The following is an example of connecting two Intranets using an encrypted PPTP tunnel over the Internet.



There are two routers in this example:

- [HomeOffice]

  Interface LocalHomeOffice 10.150.2.254/24

  Interface ToInternet 192.168.80.1/24

- [RemoteOffice]

  Interface ToInternet 192.168.81.1/24

  Interface LocalRemoteOffice 10.150.1.254/24

Each router is connected to a different ISP. One router can access another router through the Internet.

On the `Preforma` PPTP server a user must be set up for the client:

```
[admin@HomeOffice] ppp secret> add name=ex service=pptp password=lkjrht
local-address=10.0.103.1 remote-address=10.0.103.2
[admin@HomeOffice] ppp secret> print detail
Flags: X - disabled
  0   name="ex" service=pptp caller-id="" password="lkjrht" profile=default
      local-address=10.0.103.1 remote-address=10.0.103.2 routes==""
```

```
[admin@HomeOffice] ppp secret>
```

Then the user should be added in the PPTP server list:

```
[admin@HomeOffice] interface pptp-server> add user=ex
[admin@HomeOffice] interface pptp-server> print
Flags: X - disabled, D - dynamic, R - running
  #     NAME                USER         MTU   CLIENT-ADDRESS  UPTIME
ENC...
  0     pptp-in1            ex
[admin@HomeOffice] interface pptp-server>
```

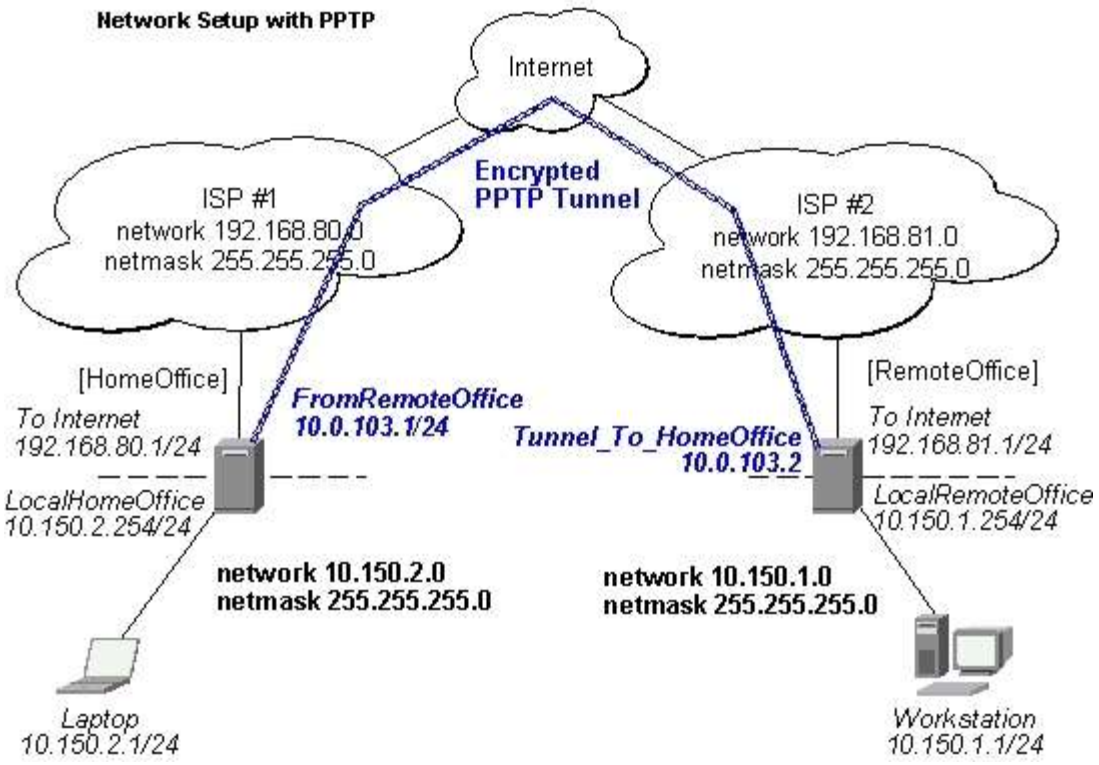
And finally, the server must be enabled:

```
[admin@HomeOffice] interface pptp-server server> set enabled=yes
[admin@HomeOffice] interface pptp-server server> print
            enabled: yes
                mtu: 1460
                mru: 1460
     authentication: mschap2
    default-profile: default
[admin@HomeOffice] interface pptp-server server>
```

Add a PPTP client to the RemoteOffice router:

```
[admin@RemoteOffice] interface pptp-client> add connect-to=192.168.80.1
user=ex \
\... password=lkjrht disabled=no
[admin@RemoteOffice] interface pptp-client> print
Flags: X - disabled, R - running
  0  R name="pptp-out1" mtu=1460 mru=1460 connect-to=192.168.80.1 user="ex"
       password="lkjrht" profile=default add-default-route=no


[admin@RemoteOffice] interface pptp-client>
```

Thus, a PPTP tunnel is created between the routers. This tunnel is like an Ethernet point-to-point connection between the routers with IP addresses 10.0.103.1 and 10.0.103.2 at each router. It enables 'direct' communication between the routers over third party networks.



To route the local Intranets over the PPTP tunnel you need to add these routes:

```
[admin@HomeOffice] > ip route add dst-address 10.150.1.0/24 gateway 10.0.103.2
[admin@RemoteOffice] > ip route add dst-address 10.150.2.0/24 gateway
10.0.103.1
```

On the PPTP server it can alternatively be done using **routes** parameter of the user configuration:

```
[admin@HomeOffice] ppp secret> print detail
Flags: X - disabled
  0   name="ex" service=pptp caller-id="" password="lkjrht" profile=default
      local-address=10.0.103.1 remote-address=10.0.103.2 routes==""

[admin@HomeOffice] ppp secret> set 0 routes="10.150.1.0/24 10.0.103.2 1"
[admin@HomeOffice] ppp secret> print detail
Flags: X - disabled
  0   name="ex" service=pptp caller-id="" password="lkjrht" profile=default
      local-address=10.0.103.1 remote-address=10.0.103.2
      routes="10.150.1.0/24 10.0.103.2 1"

[admin@HomeOffice] ppp secret>
```

Test the PPTP tunnel connection:

```
[admin@RemoteOffice]> /ping 10.0.103.1
10.0.103.1 pong: ttl=255 time=3 ms
10.0.103.1 pong: ttl=255 time=3 ms
10.0.103.1 pong: ttl=255 time=3 ms
ping interrupted
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 3/3.0/3 ms
```

Test the connection through the PPTP tunnel to the LocalHomeOffice interface:

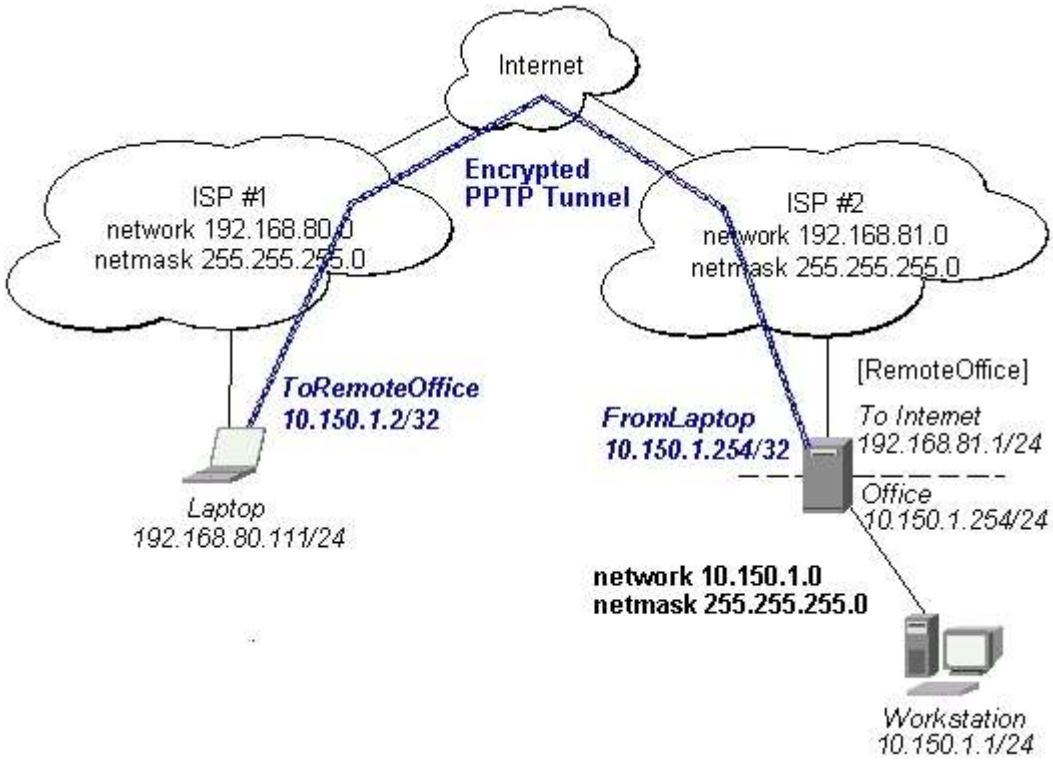
```
[admin@RemoteOffice]> /ping 10.150.2.254
10.150.2.254 pong: ttl=255 time=3 ms
10.150.2.254 pong: ttl=255 time=3 ms
10.150.2.254 pong: ttl=255 time=3 ms
ping interrupted
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 3/3.0/3 ms
```

To bridge a LAN over this secure tunnel, please see the example in the 'EoIP' section of the manual. To set the maximum speed for traffic over this tunnel, please consult the 'Queues' section.

Connecting a Remote Client via PPTP Tunnel

The following example shows how to connect a computer to a remote office network over PPTP encrypted tunnel giving that computer an IP address from the same network as the remote office has (without need of bridging over EoIP tunnels)

Please, consult the respective manual on how to set up a PPTP client with the software You are using.



The router in this example:

- [RemoteOffice]

  Interface ToInternet 192.168.81.1/24

  Interface Office 10.150.1.254/24

The client computer can access the router through the Internet.

On the PPTP server a user must be set up for the client:

```
[admin@RemoteOffice] ppp secret> add name=ex service=pptp password=lkjrht
local-address=10.150.1.254 remote-address=10.150.1.2
[admin@RemoteOffice] ppp secret> print detail
Flags: X - disabled
  0   name="ex" service=pptp caller-id="" password="lkjrht" profile=default
      local-address=10.150.1.254 remote-address=10.150.1.2 routes==""

[admin@RemoteOffice] ppp secret>
```

Then the user should be added in the PPTP server list:

```
[admin@RemoteOffice] interface pptp-server> add name=FromLaptop user=ex
```

```
[admin@RemoteOffice] interface pptp-server> print
Flags: X - disabled, D - dynamic, R - running
  #     NAME                USER         MTU   CLIENT-ADDRESS  UPTIME
ENC...
  0     FromLaptop          ex
[admin@RemoteOffice] interface pptp-server>
```

And the server must be enabled:

```
[admin@RemoteOffice] interface pptp-server server> set enabled=yes
[admin@RemoteOffice] interface pptp-server server> print
            enabled: yes
                mtu: 1460
                mru: 1460
     authentication: mschap2
    default-profile: default
[admin@RemoteOffice] interface pptp-server server>
```

Finally, the proxy APR must be enabled on the 'Office' interface:

```
[admin@RemoteOffice] interface ethernet> set Office arp=proxy-arp
[admin@RemoteOffice] interface ethernet> print
Flags: X - disabled, R - running
  #    NAME                MTU   MAC-ADDRESS       ARP
  0  R ToInternet          1500  00:30:4F:0B:7B:C1 enabled
  1  R Office              1500  00:30:4F:06:62:12 proxy-arp
[admin@RemoteOffice] interface ethernet>
```

PPTP Setup for Windows

Microsoft provides PPTP client support for Windows NT, 2000, ME, 98SE, and 98. Windows 98SE, 2000, and ME include support in the Windows setup or automatically install PPTP. For 95, NT, and 98, installation requires a download from Microsoft. Many ISPs have made help pages to assist clients with Windows PPTP installation.

- http://www.real-time.com/Customer_Support/PPTP_Config/pptp_config.html
- http://www.microsoft.com/windows95/downloads/contents/WUAdminTools/S_WUNetworkingTools/W95WinsockUpgrade/Default.asp

Sample instructions for PPTP (VPN) installation and client setup - Windows 98SE

If the VPN (PPTP) support is installed, select 'Dial-up Networking' and 'Create a new connection'. The option to create a 'VPN' should be selected. If there is no 'VPN' options, then follow the installation instructions below. When asked for the 'Host name or IP address of the VPN server', type the IP address of the router. Double-

click on the 'new' icon and type the correct user name and password (must also be in the user database on the router or RADIUS server used for authentication).

The setup of the connections takes nine seconds after selection the 'connect' button. It is suggested that the connection properties be edited so that 'NetBEUI', 'IPX/SPX compatible', and 'Log on to network' are unselected. The setup time for the connection will then be two seconds after the 'connect' button is selected.

To install the 'Virtual Private Networking' support for Windows 98SE, go to the 'Setting' menu from the main 'Start' menu. Select 'Control Panel', select 'Add/Remove Program', select the 'Windows setup' tab, select the 'Communications' software for installation and 'Details'. Go to the bottom of the list of software and select 'Virtual Private Networking' to be installed.

Troubleshooting

Description

- **I use firewall and I cannot establish PPTP connection**

  Make sure the TCP connections to port 1723 can pass through both directions between your sites. Also, IP protocol 47 should be passed through

HauteRouterOS RADIUS client

Document revision: 0.4 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

This document provides information about RouterOS built-in RADIUS client configuration, supported RADIUS attributes and recommendations on RADIUS server selection.

Specifications

Packages required: ***system***
License required: *Level1*
Submenu level: ***/radius***
Standards and Technologies: RADIUS

Description

RADIUS, short for Remote Authentication Dial-In User Service, is a remote server that provides authentication and accounting facilities to various network apliances. RADIUS authentication and accounting gives the ISP or network administrator ability to manage PPP user access and accounting from one server throughout a large network. The HauteRouterOS has a RADIUS client which can authenticate for HotSpot, PPP, PPPoE, PPTP, L2TP and ISDN connections. The attributes received from RADIUS server override the ones set in the default profile, but if some parameters are not received they are taken from the respective default profile.

The RADIUS server database is consulted only if no matching user acces record is found in router's local database.

Traffic is accounted locally with Hautespot Traffic Flow and Cisco **IP pairs** and snapshot image can be gathered using Syslog utilities. If RADIUS accounting is enabled, accounting information is also sent to the RADIUS server default for that service.

RADIUS Client Setup

Submenu level: ***/radius***

Description

This facility allows you to set RADIUS servers the router will use to authenticate users.

Property Description

accounting-backup (yes | no; default: **no**) - this entry is a backup RADIUS accounting server
accounting-port (*integer*; default: **1813**) - RADIUS server port used for accounting
address (*IP address*; default: **0.0.0.0**) - IP address of the RADIUS server
authentication-port (*integer*; default: **1812**) - RADIUS server port used for authentication
called-id (*text*; default: **""**) - value depends on Point-to-Point protocol:
ISDN - phone number dialled (MSN)
PPPoE - service name
PPTP - server's IP address
L2TP - server's IP address
domain (*text*; default: **""**) - Microsoft Windows domain of client passed to RADIUS servers that require domain validation
realm (*text*) - explicitly stated realm (user domain), so the users do not have to provide proper ISP domain name in user name
secret (*text*; default: **""**) - shared secret used to access the RADIUS server
service (*multiple choice:* hotspot | login | ppp | telephony | wireless | dhcp; default: **""**) - router services that will use this RADIUS server

hotspot - HotSpot authentication service
login - router's local user authentication
ppp - Point-to-Point clients authentication
telephony - IP telephony accounting
wireless - wireless client authentication (client's MAC address is sent as **User-Name**)
dhcp - DHCP protocol client authentication (client's MAC address is sent as **User-Name**)
timeout (*time*; default: **100ms**) - timeout after which the request should be resend

Notes

The order of the items in this list is significant.

Microsoft Windows clients send their usernames in form **domain\username**

When RADIUS server is authenticating user with CHAP, MS-CHAPv1, MS-CHAPv2, it is not using shared secret, secret is used only in authentication reply, and router is verifying it. So if you have wrong shared secret, RADIUS server will accept request, but router won't accept reply. You can see that with **/radius monitor** command, "bad-replies" number should increase whenever somebody tries to connect.

Example

To set a RADIUS server for **HotSpot** and **PPP** services that has **10.0.0.3** IP address and **ex** shared secret, you need to do the following:

```
[admin@Hautespot] radius> add service=hotspot,ppp address=10.0.0.3 secret=ex
[admin@Hautespot] radius> print
Flags: X - disabled
  #   SERVICE         CALLED-ID     DOMAIN        ADDRESS         SECRET
  0   ppp,hotspot                                 10.0.0.3        ex
[admin@Hautespot] radius>
AAA for the respective services should be enabled too:
[admin@Hautespot] radius> /ppp aaa set use-radius=yes
[admin@Hautespot] radius> /ip hotspot profile set default use-radius=yes
To view some statistics for a client:
[admin@Hautespot] radius> monitor 0
             pending: 0
            requests: 10
             accepts: 4
             rejects: 1
             resends: 15
            timeouts: 5
         bad-replies: 0
    last-request-rtt: 0s
[admin@Hautespot] radius>
```

Connection Terminating from RADIUS

Submenu level: ***/radius incoming***

Description

This facility supports unsolicited messages sent from RADIUS server. Unsolicited messages extend RADIUS protocol commands, that allow to terminate a session which has already been connected from RADIUS server. For this purpose DM (Disconnect-Messages) are used. Disconnect messages cause a user session to be terminated immediately

Property Description

accept (yes | no; default: **no**) - Whether to accept the unsolicited messages
port (*integer*; default: **1700**) - The port number to listen for the requests on

Notes

RouterOS doesn't support POD (Packet of Disconnect) the other RADIUS access request packet that performs a similar function as Disconnect Messages

Suggested RADIUS Servers

Description

HauteRouterOS RADIUS Client should work well with all RFC compliant servers. It has been tested with:

- FreeRADIUS
- XTRadius (does not currently support MS-CHAP)
- Steel-Belted Radius

Supported RADIUS Attributes

Description

Hautespot RADIUS Dictionaries

Here you can download Hautespot reference dictionary, which incorporates all the needed RADIUS attributes. This dictionary is the minimal dictionary, which is enough to support all features of Hautespot RouterOS. It is designed for FreeRADIUS, but may also be used with many other UNIX RADIUS servers (eg. XTRadius).

Note that it may conflict with the default configuration files of RADIUS server, which have references to the Attributes, absent in this dictionary. Please correct the configuration files, not the dictionary, as no other Attributes are supported by Hautespot RouterOS.

There is also dictionary.Hautespot that can be included in an existing dictionary to support Hautespot vendor-specific Attributes.

Definitions

- **PPPs** - PPP, PPTP, PPPoE and ISDN
- **default configuration** - settings in default profile (for PPPs) or HotSpot server settings (for HotSpot)

Access-Request

- **Service-Type** - always is "Framed" (only for PPPs)
- **Framed-Protocol** - always is "PPP" (only for PPPs)
- **NAS-Identifier** - router identity
- **NAS-IP-Address** - IP address of the router itself
- **NAS-Port** - unique session ID
- **NAS-Port-Type** - async PPP - "Async"; PPTP and L2TP - "Virtual"; PPPoE - "Ethernet"; ISDN - "ISDN Sync"; HotSpot - "Ethernet | Cable | Wireless-802.11" (according to the value of **nas-port-type** parameter in **/ip hotspot profile**
- **Calling-Station-Id** - PPPoE - client MAC address in capital letters; PPTP and L2TP - client public IP address; HotSpot - MAC address of the client if it is known, or IP address of the client if MAC address is unknown; ISDN - client MSN
- **Called-Station-Id** - PPPoE - service name; PPTP and L2TP - server IP address; ISDN - interface MSN; HotSpot - MAC of the hotspot interface (if known), else IP of hotspot interface specified in hotspot menu (if set), else attribute not present
- **NAS-Port-Id** - async PPP - serial port name; PPPoE - ethernet interface name on which server is running; HotSpot - name of the hotspot interface (if known), otherwise - not present; not present for ISDN, PPTP and L2TP
- **Framed-IP-Address** - IP address of HotSpot client after Universal Client translation
- **Host-IP** - IP address of HotSpot client before Universal Client translation (the original IP address of the client)
- **User-Name** - client login name
- **MS-CHAP-Domain** - User domain, if present
- **Realm** - If it is set in **/radius** menu, it is included in every RADIUS request as **Hautespot-Realm** attribute. If it is not set, the same value is sent as in **MS-CHAP-Domain** attribute (if **MS-CHAP-Domain** is missing, **Realm** is not included neither)

Depending on authentication methods (NOTE: HotSpot uses CHAP by default and may use also PAP if unencrypted passwords are enabled, it can not use MSCHAP):

- **User-Password** - encrypted password (used with PAP authentication)
- **CHAP-Password, CHAP-Challenge** - encrypted password and challenge (used with CHAP authentication)
- **MS-CHAP-Response, MS-CHAP-Challenge** - encrypted password and challenge (used with MS-CHAPv1 authentication)

- **MS-CHAP2-Response, MS-CHAP-Challenge** - encrypted password and challenge (used with MS-CHAPv2 authentication)

Access-Accept

- **Framed-IP-Address** - IP address given to client. PPPs - if address belongs to 127.0.0.0/8 or 224.0.0.0/3 networks, IP pool is used from the default profile to allocate client IP address. HotSpot - used only for dhcp-pool login method (ignored for enabled-address method), if address is 255.255.255.254, IP pool is used from HotSpot settings; if Framed-IP-Address is specified, Framed-Pool is ignored
- **Framed-IP-Netmask** - client netmask. PPPs - if specified, a route will be created to the network Framed-IP-Address belongs to via the Framed-IP-Address gateway; HotSpot - ignored by HotSpot
- **Framed-Pool** - IP pool name (on the router) from which to get IP address for the client. If specified, overrides Framed-IP-Address

NOTE: if Framed-IP-Address or Framed-Pool is specified it overrides remote-address in default configuration

- **Idle-Timeout** - overrides idle-timeout in the default configuration
- **Session-Timeout** - overrides session-timeout in the default configuration
- **Max-Session-Time** - maximum session length (uptime) the user is allowed to
- **Class** - cookie, will be included in Accounting-Request unchanged
- **Framed-Route** - routes to add on the server. Format is specified in RFC2865 (Ch. 5.22), can be specified as many times as needed
- **Filter-Id** - firewall filter chain name. It is used to make a dynamic firewall rule. Firewall chain name can have suffix .in or .out, that will install rule only for incoming or outgoing traffic. Multiple Filter-id can be provided, but only last ones for incoming and outgoing is used. For PPPs - filter rules in **ppp** chain that will jump to the specified chain, if a packet has come to/from the client (that means that you should first create a **ppp** chain and make **jump** rules that would put actual traffic to this chain). The same applies for HotSpot, but the rules will be created in **hotspot** chain
- **Mark-Id** - firewall mangle chain name (HotSpot only). The Hautespot RADIUS client upon receiving this attribute creates a dynamic firewall mangle rule with **action=jump chain=hotspot** and **jump-target** equal to the atribute value. Mangle chain name can have suffixes .in or .out, that will install rule only for incoming or outgoing traffic. Multiple Mark-id attributes can be provided, but only last ones for incoming and outgoing is used.
- **Acct-Interim-Interval** - interim-update for RADIUS client, if 0 uses the one specified in RADIUS client
- **MS-MPPE-Encryption-Policy** - require-encryption property (PPPs only)
- **MS-MPPE-Encryption-Types** - use-encryption property, non-zero value means to use encryption (PPPs only)
- **Ascend-Data-Rate** - tx/rx data rate limitation if multiple attributes are provided, first limits tx data rate, second - rx data rate. If used together with **Ascend-Xmit-Rate**, specifies rx rate. 0 if unlimited

- **Ascend-Xmit-Rate** - tx data rate limitation. It may be used to specify tx limit only instead of sending two sequental **Ascend-Data-Rate** attributes (in that case **Ascend-Data-Rate** will specify the receive rate). 0 if unlimited
- **MS-CHAP2-Success** - auth. response if MS-CHAPv2 was used (for PPPs only)
- **MS-MPPE-Send-Key, MS-MPPE-Recv-Key** - encryption keys for encrypted PPPs provided by RADIUS server only is MS-CHAPv2 was used as authentication (for PPPs only)
- **Ascend-Client-Gateway** - client gateway for DHCP-pool HotSpot login method (HotSpot only)
- **Recv-Limit** - total receive limit in bytes for the client
- **Xmit-Limit** - total transmit limit in bytes for the client
- **Wireless-Forward** - not forward the client's frames back to the wireless infrastructure if this attribute is set to "0" (Wireless only)
- **Wireless-Skip-Dot1x** - disable 802.1x authentication for the particulat wireless client if set to non-zero value (Wireless only)
- **Wireless-Enc-Algo** - WEP encryption algorithm: 0 - no encryption, 1 - 40-bit WEP, 2 - 104-bit WEP (Wireless only)
- **Wireless-Enc-Key** - WEP encruption key for the client (Wireless only)
- **Rate-Limit** - Datarate limitation for clients. Format is: **rx-rate[/tx-rate] [rx-burst-rate[/tx-burst-rate] [rx-burst-threshold[/tx-burst-threshold] [rx-burst-time[/tx-burst-time] [priority] [rx-rate-min[/tx-rate-min]]]]** from the point of view of the router (so "rx" is client upload, and "tx" is client download). All rates should be numbers with optional 'k' (1,000s) or 'M' (1,000,000s). If tx-rate is not specified, rx-rate is as tx-rate too. Same goes for tx-burst-rate and tx-burst-threshold and tx-burst-time. If both rx-burst-threshold and tx-burst-threshold are not specified (but burst-rate is specified), rx-rate and tx-rate is used as burst thresholds. If both rx-burst-time and tx-burst-time are not specified, 1s is used as default. Priority takes values 1..8, where 1 implies the highest priority, but 8 - the lowest. If rx-rate-min and tx-rate-min are not specified rx-rate and tx-rate values are used. The rx-rate-min and tx-rate-min values can not exceed rx-rate and tx-rate values.
- **Group** - Router local user group name (defines in **/user group**) for local users. HotSpot default profile for HotSpot users.
- **Advertise-URL** - URL of the page with advertisements that should be displayed to clients. If this attribute is specified, advertisements are enabled automatically, including transparent proxy, even if they were explicitly disabled in the corresponding user profile. Multiple attribute instances may be send by RADIUS server to specify additional URLs which are choosen in round robin fashion.
- **Advertise-Interval** - Time interval between two adjacent advertisements. Multiple attribute instances may be send by RADIUS server to specify additional intervals. All interval values are threated as a list and are taken one-by-one for each successful advertisement. If end of list is reached, the last value is continued to be used.

Note that the received attributes override the default ones (set in the default profile), but if an attribute is not received from RADIUS server, the default one is to be used.

Here are some Rate-Limit examples:

- **128k** - rx-rate=128000, tx-rate=128000 (no bursts)
- **64k/128M** - rx-rate=64000, tx-rate=128000000
- **64k 256k** - rx/tx-rate=64000, rx/tx-burst-rate=256000, rx/tx-burst-threshold=64000, rx/tx-burst-time=1s
- **64k/64k 256k/256k 128k/128k 10/10** - rx/tx-rate=64000, rx/tx-burst-rate=256000, rx/tx-burst-threshold=128000, rx/tx-burst-time=10s

Accounting-Request

- **Acct-Status-Type** - Start, Stop, or Interim-Update
- **Acct-Session-Id** - accounting session ID
- **Service-Type** - same as in request (PPPs only)
- **Framed-Protocol** - same as in request (PPPs only)
- **NAS-Identifier** - same as in request
- **NAS-IP-Address** - same as in request
- **User-Name** - same as in request
- **MS-CHAP-Domain** - same as in request (only for PPPs)
- **NAS-Port-Type** - same as in request
- **NAS-Port** - same as in request
- **NAS-Port-Id** - same as in request
- **Calling-Station-Id** - same as in request
- **Called-Station-Id** - same as in request
- **Acct-Authentic** - either authenticated by the RADIUS or Local authority (PPPs only)
- **Framed-IP-Address** - IP address given to the user
- **Framed-IP-Netmask** - same as in RADIUS reply
- **Class** - RADIUS server cookie (PPPs only)
- **Acct-Delay-Time** - how long does the router try to send this Accounting-Request packet

Stop and Interim-Update Accounting-Request

- **Acct-Session-Time** - connection uptime in seconds
- **Acct-Input-Octets** - bytes received from the client
- **Acct-Input-Gigawords** - 4G (2^32) bytes received from the client (bits 32..63, when bits 0..31 are delivered in **Acct-Input-Octets**) (HotSpot only)
- **Acct-Input-Packets** - nubmer of packets received from the client
- **Acct-Output-Octets** - bytes sent to the client
- **Acct-Output-Gigawords** - 4G (2^32) bytes sent to the client (bits 32..63, when bits 0..31 are delivered in **Acct-Output-Octets**) (HotSpot only)
- **Acct-Output-Packets** - number of packets sent to the client

Stop Accounting-Request

These packets can additionally have:

- **Acct-Terminate-Cause** - session termination cause (see RFC2866 ch. 5.10)

Attribute Numeric Values

Name	VendorID	Value	RFC where it is defined
Acct-Authentic		45	RFC2866
Acct-Delay-Time		41	RFC2866
Acct-Input-Gigawords		52	RFC2869
Acct-Input-Octets		42	RFC2866
Acct-Input-Packets		47	RFC2866
Acct-Interim-Interval		85	RFC2869
Acct-Output-Gigawords		53	RFC2869
Acct-Output-Octets		43	RFC2866
Acct-Output-Packets		48	RFC2866
Acct-Session-Id		44	RFC2866
Acct-Session-Time		46	RFC2866
Acct-Status-Type		40	RFC2866
Acct-Terminate-Cause		49	RFC2866
Ascend-Client-Gateway	529	132	
Ascend-Data-Rate	529	197	
Ascend-Xmit-Rate	529	255	
Called-Station-Id		30	RFC2865
Calling-Station-Id		31	RFC2865
CHAP-Challenge		60	RFC2866
CHAP-Password		3	RFC2865
Class		25	RFC2865
Filter-Id		11	RFC2865
Framed-IP-Address		8	RFC2865
Framed-IP-Netmask		9	RFC2865
Framed-Pool		88	RFC2869
Framed-Protocol		7	RFC2865

Framed-Route		22	RFC2865
Group	14988	3	
Idle-Timeout		28	RFC2865
MS-CHAP-Challenge	311	11	RFC2548
MS-CHAP-Domain	311	10	RFC2548
MS-CHAP-Response	311	1	RFC2548
MS-CHAP2-Response	311	25	RFC2548
MS-CHAP2-Success	311	26	RFC2548
MS-MPPE-Encryption-Policy	311	7	RFC2548
MS-MPPE-Encryption-Types	311	8	RFC2548
MS-MPPE-Recv-Key	311	17	RFC2548
MS-MPPE-Send-Key	311	16	RFC2548
NAS-Identifier		32	RFC2865
NAS-Port		5	RFC2865
NAS-Port-Id		87	RFC2869
NAS-Port-Type		61	RFC2865
Rate-Limit	14988	8	
Realm	14988	9	
Recv-Limit	14988	1	
Service-Type		6	RFC2865
Session-Timeout		27	RFC2865
User-Name		1	RFC2865
User-Password		2	RFC2865
Wireless-Enc-Algo	14988	6	
Wireless-Enc-Key	14988	7	
Wireless-Forward	14988	4	
Wireless-Skip-Dot1x	14988	5	

Xmit-Limit	14988	2	

Troubleshooting

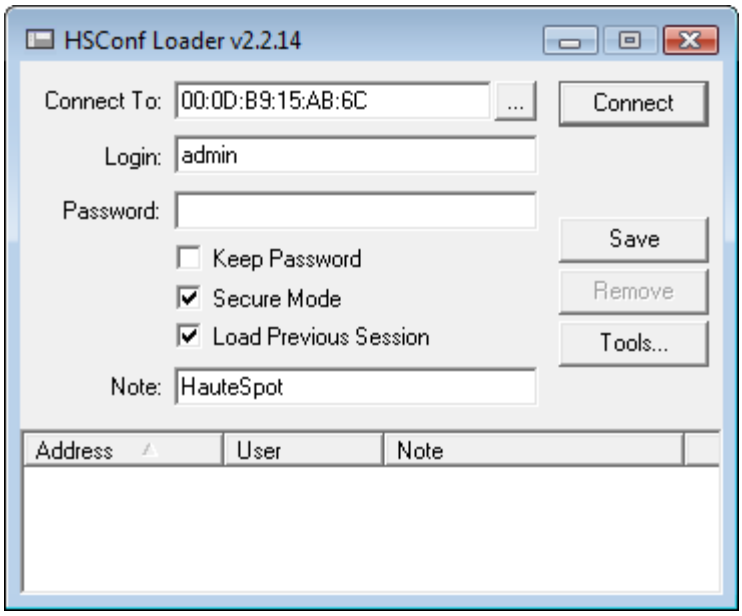
Description

- **My radius server accepts authentication request from the client with "Auth: Login OK:...", but the user cannot log on. The bad replies counter is incrementing under radius monitor**

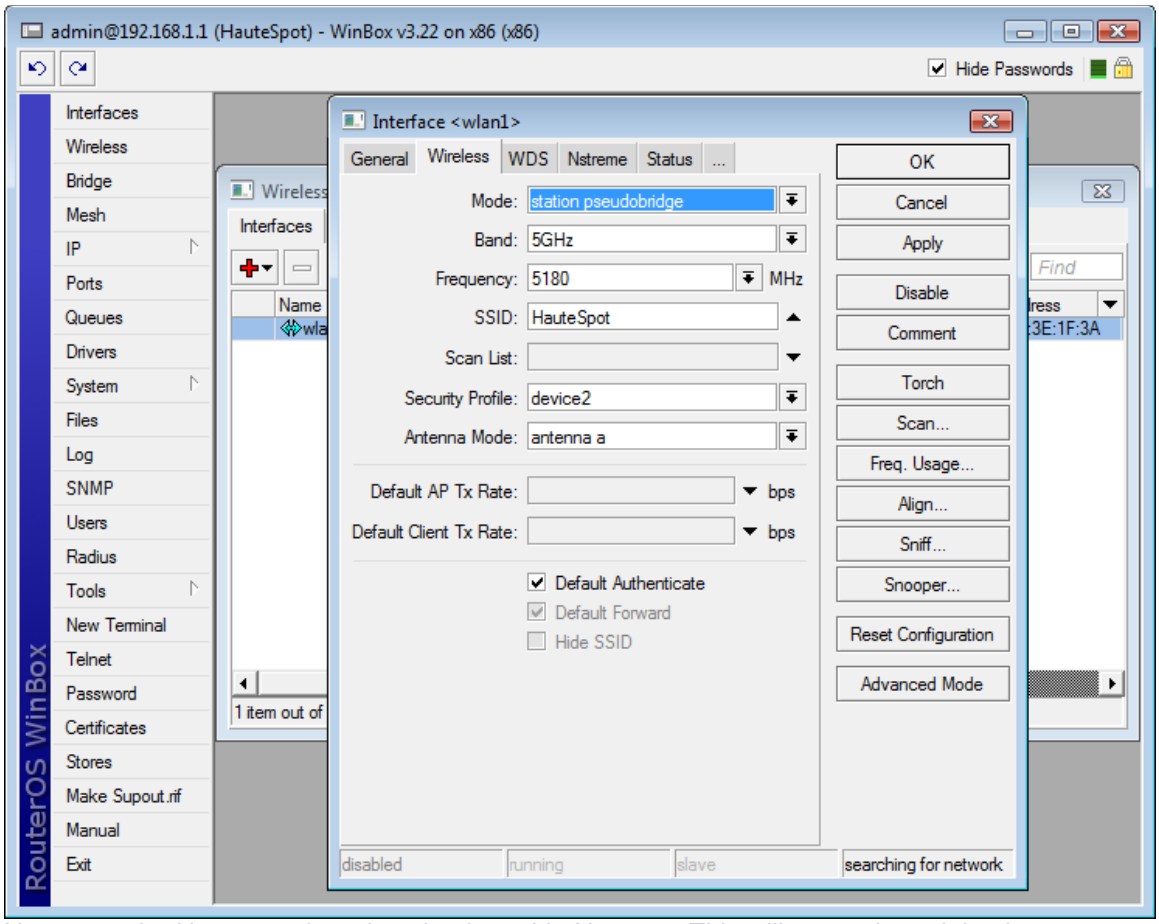
  This situation can occur, if the radius client and server have high delay link between them. Try to increase the radius client's timeout to 600ms or more instead of the default 300ms! Also, double check, if the secrets match on client and server!

Configuring the HauteWRAP family from HauteSpot TDMA-like protocol (TLP) mode to 802.11

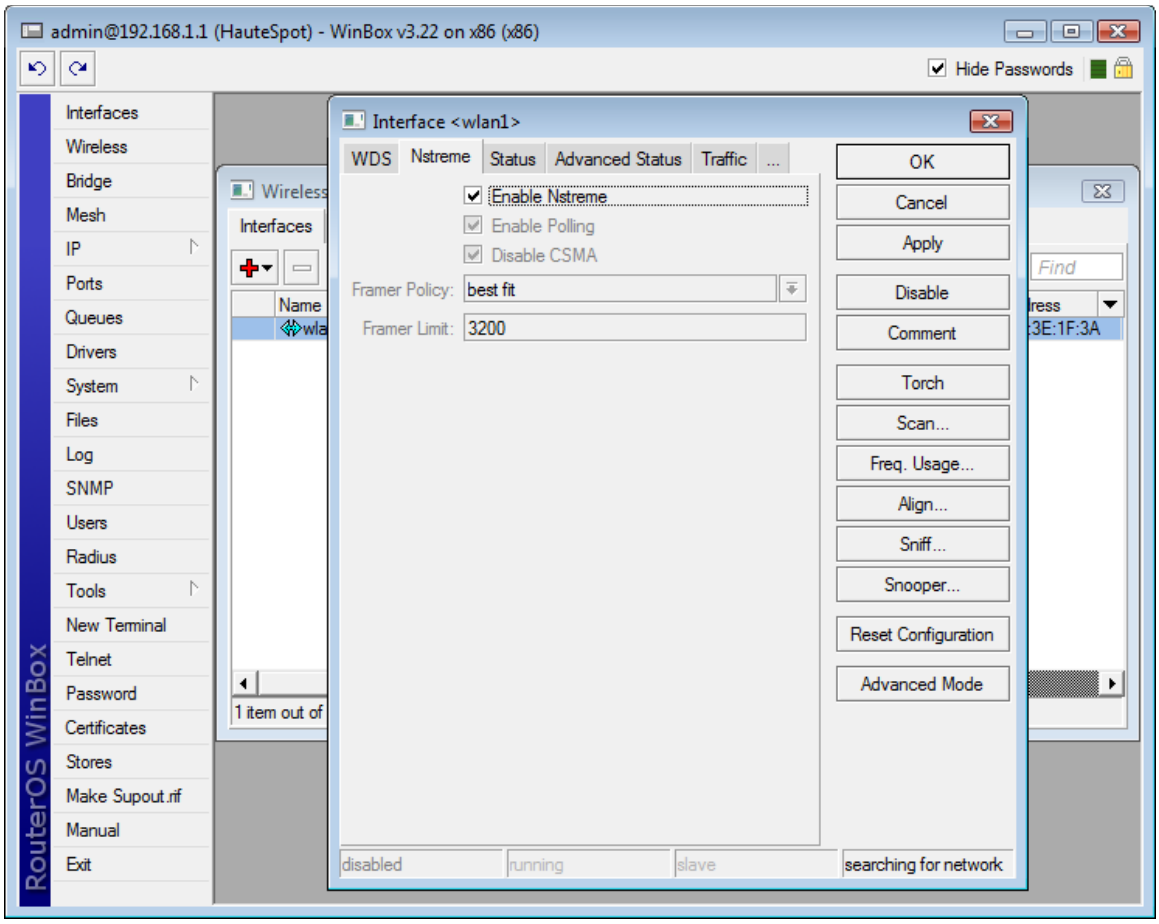
The HauteWRAP family of products is intended to support video streaming applications by default. This means that the routers come configured with TLP enabled by default. TLP is a wireless protocol designed for video and uses TDMA type polling in order to provide low latency, low jitter and even distribution of bandwidth. The result is the best performing wireless network for video available.
However, many of our customers prefer to operate in standard 802.11 or mesh modes. It is very simple to convert your HauteWRAP to 802.11 or mesh modes.
First, open a connection to your HauteWRAP using the HSCONF utility. If you are not familiar with HSCONF, refer to the user manual.



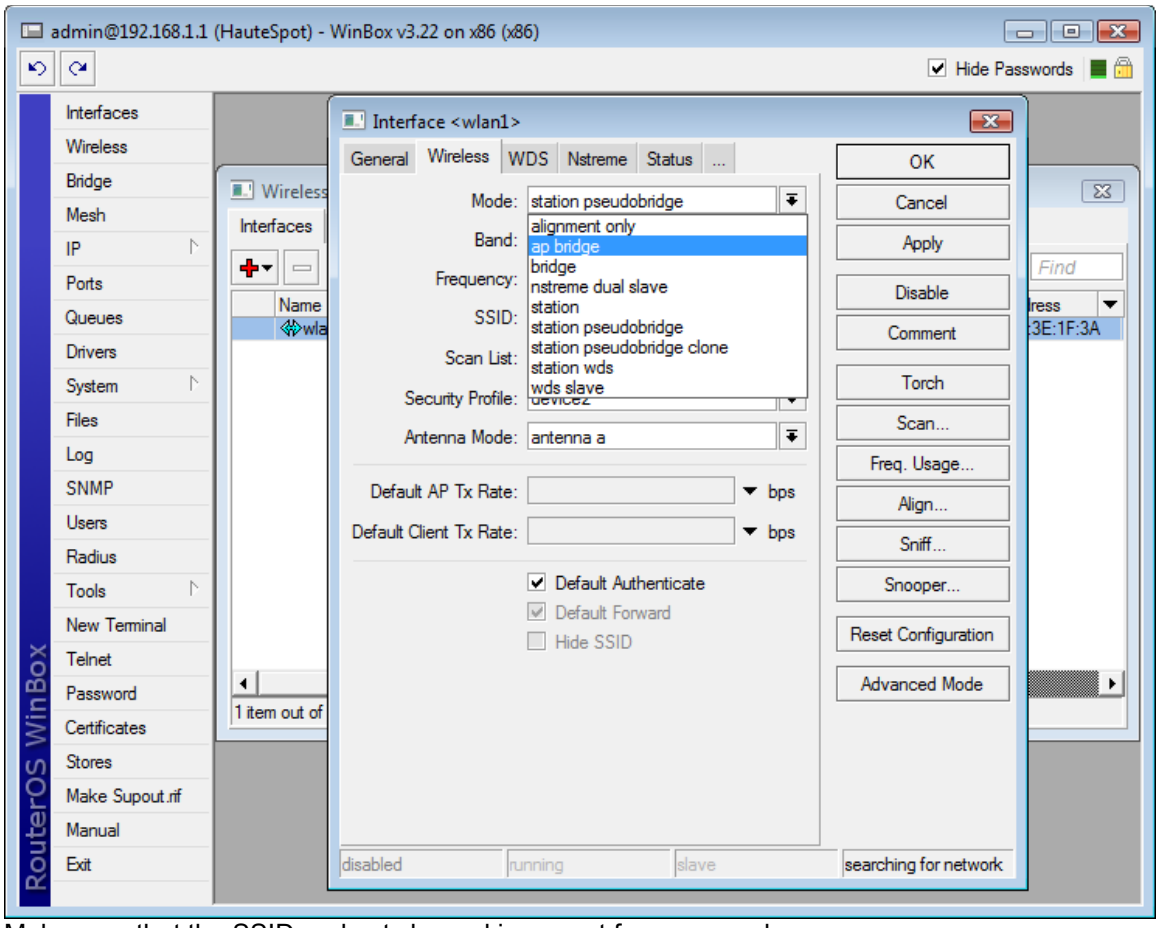
Either connect using the MAC address or IP address. If you are unsure of the MAC address you can use the ... to search available HauteWRAP units. This will display a list of available units. Simply click on either the MAC address or IP address to connect. Then hit the connect button. Once connected, open the Wireless dialog and select the wireless interface which you wish to enable 802.11 for.



Now go to the Nstreme tab and uncheck enable Nstreme. This will revert the unit back to standard 802.11 mode. Please remember that the system is operating in 802.11 mode when configuring the unit using the web interface, as the web interface will still indicate that the unit is in TDMA(TLP) operating mode.



If you wish to set the unit to be an Access Point, then the only other change required is to set the operating mode to be Access Point/Bridge on the wireless tab:



Make sure that the SSID and set channel is correct for your needs.

Once completed, the system is ready to be used in 802.11 mode.

For mesh operation, refer to the HauteSpot HWMP manual.

HauteRouterOS RIP

Document revision: 1 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

HauteRouterOS implements RIP Version 1 (RFC1058) and Version 2 (RFC 2453). RIP enables routers in an autonomous system to exchange routing information. It always uses the best path (the path with the fewest number of hops (i.e. routers)) available.

Specifications

Packages required: ***routing***
License required: *Level3*
Submenu level: ***/routing rip***
Standards and Technologies: RIPv1, RIPv2
Hardware usage: *Not significant*

Description

Routing Information Protocol (RIP) is one protocol in a series of routing protocols based on Bellman-Ford (or distance vector) algorithm. This Interior Gateway Protocol (IGP) lets routers exchange routing information across a single autonomous system in the way of periodic RIP updates. Routers transmit their own RIP updates to neighboring networks and listen to the RIP updates from the routers on those neighboring networks to ensure their routing table reflects the current state of the network and all the best paths are available. Best path considered to be a path with the fewest hop count (*id est* that include fewer routers).

The routes learned by RIP protocol are installed in the route list (**/ip route print**) with the distance of 120.

Additional Resources

- RIPv1 Protocol
- RIPv2 Protocol
- Cisco Systems RIP protocol overview

General Setup

Property Description

redistribute-static (yes | no; default: **no**) - specifies whether to redistribute static routes to neighbour routers or not
redistribute-connected (yes | no; default: **no**) - specifies whether to redistribute connected routes to neighbour routers or not
redistribute-ospf (yes | no; default: **no**) - specifies whether to redistribute routes learned via OSPF protocol to neighbour routers or not
redistribute-bgp (yes | no; default: **no**) - specifies whether to redistribute routes learned via bgp protocol to neighbour routers or not
metric-static (*integer*; default: **1**) - specifies metric (the number of hops) for the static routes
metric-connected (*integer*; default: **1**) - specifies metric (the number of hops) for the connected routes
metric-ospf (*integer*; default: **1**) - specifies metric (the number of hops) for the routes learned via OSPF protocol

metric-bgp (*integer*; default: **1**) - specifies metric (the number of hops) for the routes learned via BGP protocol
update-timer (*time*; default: **30s**) - specifies frequency of RIP updates
timeout-timer (*time*; default: **3m**) - specifies time interval after which the route is considered invalid
garbage-timer (*time*; default: **2m**) - specifies time interval after which the invalid route will be dropped from neighbor router table

Notes

The maximum metric of RIP route is **15**. Metric higher than **15** is considered 'infinity' and routes with such metric are considered unreachable. Thus RIP cannot be used on networks with more than 15 hops between any two routers, and using **redistribute** metrics larger that **1** further reduces this maximum hop count.

Example

To enable RIP protocol to redistribute the routes to the connected networks:

```
[admin@Hautespot] routing rip> set redistribute-connected=yes
[admin@Hautespot] routing rip> print
       redistribute-static: no
    redistribute-connected: yes
         redistribute-ospf: no
          redistribute-bgp: no
             metric-static: 1
          metric-connected: 1
               metric-ospf: 1
                metric-bgp: 1
              update-timer: 30s
             timeout-timer: 3m
             garbage-timer: 2m
[admin@Hautespot] routing rip>
```

Interfaces

Submenu level: ***/routing rip interface***

Description

In general you do not have to configure interfaces in order to run RIP. This command level is provided only for additional configuration of specific RIP interface parameters.

Property Description

interface (*name*; default: **all**) - interface on which RIP runs
all - sets defaults for interfaces not having any specific settings

send (v1 | v1-2 | v2; default: **v2**) - specifies RIP protocol update versions to distribute
receive (v1 | v1-2 | v2; default: **v2**) - specifies RIP protocol update versions the router will be able to receive
authentication (none | simple | md5; default: **none**) - specifies authentication method to use for RIP messages
none - no authentication performed
simple - plain text authentication
md5 - Keyed Message Digest 5 authentication
authentication-key (*text*; default: **""**) - specifies authentication key for RIP messages
prefix-list-in (*name*; default: **""**) - name of the filtering prefix list for received routes
prefix-list-out (*name*; default: **""**) - name of the filtering prefix list for advertised routes

Notes

It is recommended not to use RIP version 1 wherever it is possible due to security issues

Example

To add an entry that specifies that when advertising routes through the **ether1** interface, prefix list **plout** should be applied:

```
[admin@Hautespot] routing rip> interface add interface=ether1 \
\... prefix-list-out=plout
[admin@Hautespot] routing rip> interface print
Flags: I - inactive
  0   interface=ether1 receive=v2 send=v2 authentication=none
      authentication-key="" prefix-list-in=plout prefix-list-out=none


[admin@Hautespot] routing rip>
```

Networks

Submenu level: ***/routing rip network***

Description

To start the RIP protocol, you have to define the networks on which RIP will run.

Property Description

address (*IP address mask*; default: **0.0.0.0/0**) - specifies the network on which RIP will run. Only directly connected networks of the router may be specified
netmask (*IP address*; default: **0.0.0.0**) - specifies the network part of the address (if it is not specified in the **address** argument)

Notes

For point-to-point links you should specify the remote endpoint IP address as the network IP address. For this case the correct **netmask** will be **/32**.

Example

To enable RIP protocol on **10.10.1.0/24** network:

```
[admin@Hautespot] routing rip network> add address=10.10.1.0/24
[admin@Hautespot] routing rip network> print
  # ADDRESS
  0 10.10.1.0/24
[admin@Hautespot] routing rip>
```

Neighbors

Description

This submenu is used to define a neighboring routers to exchange routing information with. Normally there is no need to add the neighbors, if multicasting is working properly within the network. If there are problems with exchanging routing information, neighbor routers can be added to the list. It will force the router to exchange the routing information with the neighbor using regular unicast packets.

Property Description

address (*IP address*; default: **0.0.0.0**) - IP address of neighboring router

Example

To force RIP protocol to exchange routing information with the **10.0.0.1** router:

```
[admin@Hautespot] routing rip> neighbor add address=10.0.0.1
[admin@Hautespot] routing rip> neighbor print
Flags: I - inactive
  #   ADDRESS
  0   10.0.0.1
[admin@Hautespot] routing rip>
```

Routes

Submenu level: ***/routing rip route***

Property Description

dst-address (*read-only: IP address mask*) - network address and netmask of destination
gateway (*read-only: IP address*) - last gateway on the route to destination
metric (*read-only: integer*) - distance vector length to the destination network
from (*IP address*) - specifies the IP address of the router from which the route was received

Notes

This list shows routes learned by all dynamic routing protocols (RIP, OSPF and BGP)

Example

To view the list of the routes:

```
[admin@Hautespot] routing rip route> print
Flags: S - static, R - rip, O - ospf, C - connect, B - bgp
  0 O dst-address=0.0.0.0/32 gateway=10.7.1.254 metric=1 from=0.0.0.0

...

 33 R dst-address=159.148.10.104/29 gateway=10.6.1.1 metric=2 from=10.6.1.1

 34 R dst-address=159.148.10.112/28 gateway=10.6.1.1 metric=2 from=10.6.1.1

[admin@Hautespot] routing rip route>
```

Application Examples

Example

Let us consider an example of routing information exchange between Hautespot router, a Cisco router and the ISP (also Hautespot) routers:

- Hautespot Router Configuration

```
[admin@Hautespot] > interface print
```

```
Flags: X - disabled, D - dynamic, R - running
  #    NAME                TYPE             MTU
  0  R ether1              ether            1500
  1  R ether2              ether            1500
[admin@Hautespot] > ip address print
Flags: X - disabled, I - invalid, D - dynamic
  #   ADDRESS            NETWORK         BROADCAST       INTERFACE
  0   10.0.0.174/24      10.0.0.174      10.0.0.255      ether1
  1   192.168.0.1/24     192.168.0.0     192.168.0.255   ether2
[admin@Hautespot] > ip route print
Flags: X - disabled, I - invalid, D - dynamic, J - rejected,
C - connect, S - static, R - rip, O - ospf, B - bgp
     #    DST-ADDRESS        G GATEWAY         DISTANCE INTERFACE
     0 DC 192.168.0.0/24     r 0.0.0.0         0        ether2
     1 DC 10.0.0.0/24        r 0.0.0.0         0        ether1
```

```
[admin@Hautespot] >
```

Note, that no default route has been configured. The route will be obtained using the RIP. The necessary configuration of the RIP general settings is as follows:

```
[admin@Hautespot] routing rip> set redistribute-connected=yes
[admin@Hautespot] routing rip> print
       redistribute-static: no
    redistribute-connected: yes
         redistribute-ospf: no
          redistribute-bgp: no
             metric-static: 1
          metric-connected: 1
               metric-ospf: 1
                metric-bgp: 1
              update-timer: 30s
             timeout-timer: 3m
             garbage-timer: 2m

[admin@Hautespot] routing rip>
```

The minimum required configuration of RIP interface is just enabling the network associated with the ether1 interface:

```
[admin@Hautespot] routing rip network> add address=10.0.0.0/2
[admin@Hautespot] routing rip network> print
  # ADDRESS
  0 10.0.0.0/24

[admin@Hautespot] routing rip network>
```

Note, that there is no need to run RIP on the ether2, as no propagation of RIP information is required into the Remote network in this example. The routes obtained by RIP can be viewed in the /routing rip route menu:

```
[admin@Hautespot] routing rip> route print
Flags: S - static, R - rip, O - ospf, C - connect, B - bgp
  0 R dst-address=0.0.0.0/0 gateway=10.0.0.26 metric=2 from=10.0.0.26

  1 C dst-address=10.0.0.0/24 gateway=0.0.0.0 metric=1 from=0.0.0.0

  2 C dst-address=192.168.0.0/24 gateway=0.0.0.0 metric=1 from=0.0.0.0

  3 R dst-address=192.168.1.0/24 gateway=10.0.0.26 metric=1
from=10.0.0.26

  4 R dst-address=192.168.3.0/24 gateway=10.0.0.26 metric=1
from=10.0.0.26

[admin@Hautespot] routing rip>
```

The regular routing table is:

```
[Hautespot] routing rip> /ip route print
Flags: X - disabled, I - invalid, D - dynamic, J - rejected,
C - connect, S - static, R - rip, O - ospf, B - bgp
    #    DST-ADDRESS        G GATEWAY         DISTANCE INTERFACE
    0  R 0.0.0.0/0          r 10.0.0.26       120      ether1
    1  R 192.168.3.0/24     r 10.0.0.26       120      ether1
    2  R 192.168.1.0/24     r 10.0.0.26       120      ether1
    3 DC 192.168.0.0/24     r 0.0.0.0         0        ether2
    4 DC 10.0.0.0/24        r 0.0.0.0         0        ether1
[admin@Hautespot] routing rip>
```

- Cisco Router Configuration

```
Cisco#show running-config
...
interface Ethernet0
 ip address 10.0.0.26 255.255.255.0
 no ip directed-broadcast
!
interface Serial1
 ip address 192.168.1.1 255.255.255.252
 ip directed-broadcast
!
router rip
 version 2
 redistribute connected
 redistribute static
 network 10.0.0.0
 network 192.168.1.0
```

```
!
ip classless
!
```

```
...
```

The routing table of the Cisco router is:

```
Cisco#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B -
BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate
default
       U - per-user static route, o - ODR

Gateway of last resort is 192.168.1.2 to network 0.0.0.0

     10.0.0.0/24 is subnetted, 1 subnets
C       10.0.0.0 is directly connected, Ethernet0
R    192.168.0.0/24 [120/1] via 10.0.0.174, 00:00:19, Ethernet0
     192.168.1.0/30 is subnetted, 1 subnets
C       192.168.1.0 is directly connected, Serial1
R    192.168.3.0/24 [120/1] via 192.168.1.2, 00:00:05, Serial1
R*   0.0.0.0/0 [120/1] via 192.168.1.2, 00:00:05, Serial1
Cisco#
```

As we can see, the Cisco router has learned RIP routes both from the Hautespot router (192.168.0.0/24), and from the ISP router (0.0.0.0/0 and 192.168.3.0/24).

HauteRouterOS Router User AAA

Document revision: 2.3 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

This documents provides summary, configuration reference and examples on router user management.

Specifications

Packages required: ***system***
License required: *Level1*
Submenu level: ***/user***
Hardware usage: *Not significant*

Description

HauteRouterOS router user facility manage the users connecting the router from the local console, via serial terminal, telnet, SSH or HSConfig. The users are authenticated using either local database or designated RADIUS server.

Each user is assigned to a user group, which denotes the rights of this user. A group policy is a combination of individual policy items.

In case the user authentication is performed using RADIUS, the RADIUS client should be previously configured under the **/radius** submenu.

Router User Groups

Submenu level: ***/user group***

Description

The router user groups provide a convenient way to assign different permissions and access rights to different user classes.

Property Description

name (*name*) - the name of the user group
policy (*multiple choice:* local | telnet | ssh | ftp | reboot | read | write | policy | test | web; default: **!local,!telnet,!ssh,!ftp,!reboot,!read,!write,!policy,!test,!web**) - group policy item set
local - policy that grants rights to log in locally via console
telnet - policy that grants rights to log in remotely via telnet
ssh - policy that grants rights to log in remotely via secure shell protocol
ftp - policy that grants remote rights to log in remotely via FTP and to transfer files from and to the router
reboot - policy that allows rebooting the router
read - policy that grants read access to the router's configuration. All console commands that do not alter router's configuration are allowed
write - policy that grants write access to the router's configuration, except for user management. This policy does not allow to read the configuration, so make sure to enable **read** policy as well
policy - policy that grants user management rights. Should be used together with **write** policy

test - policy that grants rights to run ping, traceroute, bandwidth-test and wireless scan, sniffer and snooper commands
web - policy that grants rights to log in remotely via WebBox
HSConfig - policy that grants rights to log in remotely via HSConfig
password - policy that grants rights to change the password

Notes

There are three system groups which cannot be deleted:

```
[admin@rb13] > /user group print
 0 name="read"
policy=local,telnet,ssh,reboot,read,test,HSConfig,password,web,!ftp,!write,!po
licy

 1 name="write"
policy=local,telnet,ssh,reboot,read,write,test,HSConfig,password,web,!ftp,!pol
icy

 2 name="full"
policy=local,telnet,ssh,ftp,reboot,read,write,policy,test,HSConfig,password,we
b

 3 name="test"
policy=ssh,read,policy,!local,!telnet,!ftp,!reboot,!write,!test,!HSConfig,!pas
sword,!web
[admin@rb13] >
```

Exclamation sign '**!**' just before policy item name means **NOT**.

Example

To add **reboot** group that is allowed to reboot the router locally or using telnet, as well as read the router's configuration, enter the following command:

```
[admin@rb13] user group> add name=reboot policy=telnet,reboot,read,local
[admin@rb13] user group> print
 0 name="read"
policy=local,telnet,ssh,reboot,read,test,HSConfig,password,web,!ftp,!write,!po
licy

 1 name="write"
policy=local,telnet,ssh,reboot,read,write,test,HSConfig,password,web,!ftp,!pol
icy

 2 name="full"
policy=local,telnet,ssh,ftp,reboot,read,write,policy,test,HSConfig,password,we
b

 3 name="reboot"
policy=local,telnet,reboot,read,!ssh,!ftp,!write,!policy,!test,!HSConfig,!pass
word,!web
```

```
[admin@rb13] user group>
```

Router Users

Submenu level: ***/user***

Description

Router user database stores the information such as username, password, allowed access addresses and group about router management personnel.

Property Description

address (*IP address/netmask*; default: **0.0.0.0/0**) - host or network address from which the user is allowed to log in
group (*name*) - name of the group the user belongs to
name (*name*) - user name. Although it must start with an alphanumeric character, it may contain "*", "_", "." and "@" symbols
password (*text*; default: **""**) - user password. If not specified, it is left blank (hit [Enter] when logging in). It conforms to standard Unix characteristics of passwords and may contain letters, digits, "*" and "_" symbols

Notes

There is one predefined user with full access rights:

```
[admin@Hautespot] user> print
Flags: X - disabled
  #   NAME                                            GROUP ADDRESS
  0   ;;; system default user
      admin                                           full  0.0.0.0/0

[admin@Hautespot] user>
```

There always should be at least one user with fulls access rights. If the user with full access rights is the only one, it cannot be removed.

Example

To add user **joe** with password **j1o2e3** belonging to **write** group, enter the following command:

```
[admin@Hautespot] user> add name=joe password=j1o2e3 group=write
[admin@Hautespot] user> print
Flags: X - disabled
```

```
  0   ;;; system default user
      name="admin" group=full address=0.0.0.0/0

  1   name="joe" group=write address=0.0.0.0/0


[admin@Hautespot] user>
```

Monitoring Active Router Users

Command name: ***/user active print***

Description

This command shows the currently active users along with respective statisics information.

Property Description

address (*read-only: IP address*) - host IP address from which the user is accessing the router
0.0.0.0 - the user is logged in locally from the console
name (*read-only: name*) - user name
via (*read-only:* console | telnet | ssh | HSConfig) - user's access method
console - user is logged in locally
telnet - user is logged in remotely via telnet
ssh - user is logged in remotely via secure shell protocol
HSConfig - user is logged in remotely via HSConfig tool
when (*read-only: date*) - log in date and time

Example

To print currently active users, enter the following command:

```
[admin@rb13] user> active print
Flags: R - radius
 #   WHEN                 NAME
ADDRESS         VIA
 0   feb/27/2004 00:41:41 admin
1.1.1.200       ssh
 1   feb/27/2004 01:22:34 admin
1.1.1.200       HSConfig
[admin@rb13] user>
```

Router User Remote AAA

Submenu level: ***/user aaa***

Description

Router user remote AAA enables router user authentication and accounting via RADIUS server.

Property Description

accounting (yes | no; default: **yes**) - specifies whether to use RADIUS accounting
default-group (*name*; default: **read**) - user group used by default for users authenticated via RADIUS server
interim-update (*time*; default: **0s**) - RADIUS Interim-Update interval
use-radius (yes | no; default: **no**) - specifies whether a user database on a RADIUS server should be consulted

Notes

The RADIUS user database is consulted only if the required username is not found in the local user database

Example

To enable RADIUS AAA, enter the following command:

```
[admin@Hautespot] user aaa> set use-radius=yes
[admin@Hautespot] user aaa> print
        use-radius: yes
        accounting: yes
    interim-update: 0s
     default-group: read
[admin@Hautespot] user aaa>
```

HauteRouterOS Routes, Equal Cost Multipath Routing, Policy Routing

Document revision: 2.2 (October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

The following manual surveys the IP routes management, equal-cost multi-path (ECMP) routing technique, and policy-based routing.

Specifications

Packages required: ***system***
License required: *Level1*
Submenu level: ***/ip route***
Standards and Technologies: IP (RFC 791)
Hardware usage: *Not significant*

Description

HauteRouterOS has following types of routes:

- **dynamic routes** - automatically created routes for networks, which are directly accessed through an interface. They appear automatically, when adding a new IP address. Dynamic routes are also added by routing protocols.
- **static routes** - user-defined routes that specify the router which can forward traffic to the specified destination network. They are useful for specifying the default gateway

ECMP (Equal Cost Multi-Path) Routing

This routing mechanism enables packet routing along multiple paths with equal cost and ensures load balancing. With ECMP routing, you can use more than one gateway for one destination network (Note! This approach does not provide failover). With ECMP, a router potentially has several available next hops towards a given destination. A new gateway is chosen for each new source/destination IP pair. It means that, for example, one FTP connection will use only one link, but new connection to a different server will use another link. ECMP routing has another good feature - single connection packets do not get reordered and therefore do not kill TCP performance.

The ECMP routes can be created by routing protocols (RIP or OSPF), or by adding a static route with multiple gateways, separated by a comma (e.g., /ip route add gateway=192.168.0.1,192.168.1.1). The routing protocols may create routes (dynamic) with equal cost automatically, if the cost of the interfaces is adjusted propery. For more information on using routing protocols, please read the corresponding Manual.

Policy-Based Routing

It is a routing approach where the next hop (gateway) for a packet is chosen, based on a policy, which is configured by the network administrator. In RouterOS the procedure the follwing:

- mark the desired packets, with a **routing-mark**
- choose a gateway for the marked packets

Note! In routing process, the router decides which route it will use to send out the packet. Afterwards, when the packet is masqueraded, its source address is taken from the **prefsrc** field.

Routes

Submenu level: ***/ip route***

Description

In this submenu you can configure Static, Equal Cost Multi-Path and Policy-Based Routing and see the routes.

Property Description

as-path (*text*) - manual value of BGP's as-path for outgoing route
atomic-aggregate (yes | no) - BGP attribute. An indication to receiver that it cannot "deaggregate" the prefix
check-gateway (arp | ping; default: **ping**) - which protocol to use for gateway reachability
distance (*integer*: 0..255) - administrative distance of the route. When forwarding a packet, the router will use the route with the lowest administrative distance and reachable gateway
dst-address (*IP address*/*netmask*; default: **0.0.0.0/0**) - destination address and network mask, where netmask is number of bits which indicate network number. Used in static routing to specify the destination which can be reached, using a gateway
0.0.0.0/0 - any network
gateway (*IP address*) - gateway host, that can be reached directly through some of the interfaces. You can specify multiple gateways separated by a comma "," for ECMP routes
local-pref (*integer*) - local preference value for a route
med (*integer*) - a BGP attribute, which provides a mechanism for BGP speakers to convey to an adjacent AS the optimal entry point into the local AS
origin (incomplete | igp | egp) - the origin of the route prefix
prefsrc (*IP address*) - source IP address of packets, leaving router via this route
0.0.0.0 - prefsrc is determined automatically
prepend (*integer*: 0..16) - number which indicates how many times to prepend AS_NAME to AS_PATH
routing-mark (*name*) - a mark for packets, defined under */ip firewall mangle*. Only those packets which have the according routing-mark, will be routed, using this gateway. With this parameter we provide policy based routing
scope (*integer*: 0..255) - a value which is used to recursively lookup the nexthop addresses. Nexthop is looked up only through routes that have scope <= target-scope of the nexthop

target-scope (*integer*: 0..255) - a value which is used to recursively lookup the next-hop addresses. Each nexthop address selects smallest value of target-scope from all routes that use this nexthop address. Nexthop is looked up only through routes that have scope <= target-scope of the nexthop

Notes

You can specify more than one or two gateways in the route. Moreover, you can repeat some routes in the list several times to do a kind of cost setting for gateways.

Example

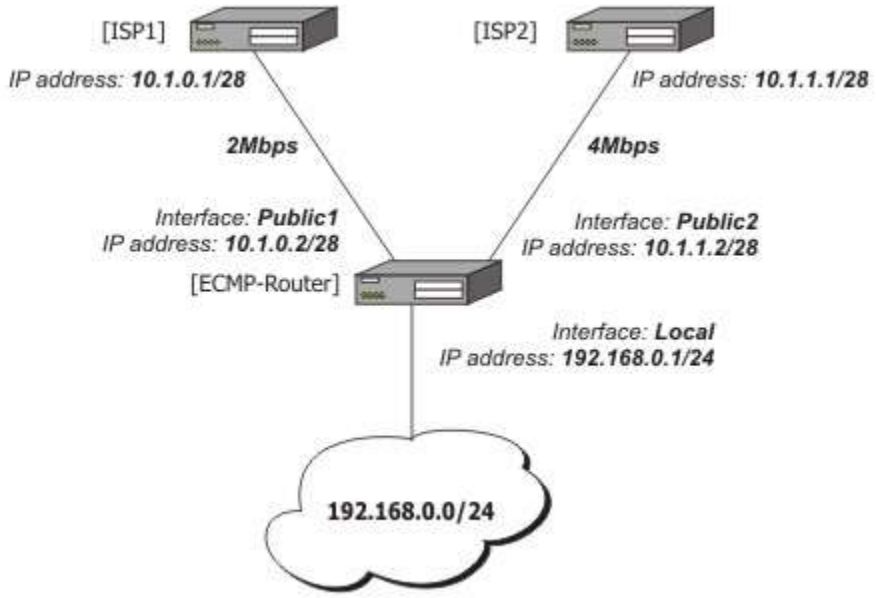
To add two static routes to networks 10.1.12.0/24 and 0.0.0.0/0 (the default destination address) on a router with two interfaces and two IP addresses:

```
[admin@Hautespot] ip route> add dst-address=10.1.12.0/24 gateway=192.168.0.253
[admin@Hautespot] ip route> add gateway=10.5.8.1
[admin@Hautespot] ip route> print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf
 #     DST-ADDRESS        G GATEWAY         DISTANCE INTERFACE
 0 A S 10.1.12.0/24       r 192.168.0.253            Local
 1 ADC 10.5.8.0/24                                   Public
 2 ADC 192.168.0.0/24                                Local
 3 A S 0.0.0.0/0          r 10.5.8.1                 Public
[admin@Hautespot] ip route>
```

Application Examples

Static Equal Cost Multi-Path routing

Consider the following situation where we have to route packets from the network **192.168.0.0/24** to 2 gateways - **10.1.0.1** and **10.1.1.1**:



Note that the ISP1 gives us 2Mbps and ISP2 - 4Mbps so we want a traffic ratio 1:2 (1/3 of the source/destination IP pairs from **192.168.0.0/24** goes through ISP1, and 2/3 through ISP2).

IP addresses of the router:

```
[admin@ECMP-Router] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
 #   ADDRESS            NETWORK         BROADCAST       INTERFACE
 0   192.168.0.254/24   192.168.0.0     192.168.0.255   Local
 1   10.1.0.2/28        10.1.0.0        10.1.0.15       Public1
 2   10.1.1.2/28        10.1.1.0        10.1.1.15       Public2
[admin@ECMP-Router] ip address>
```

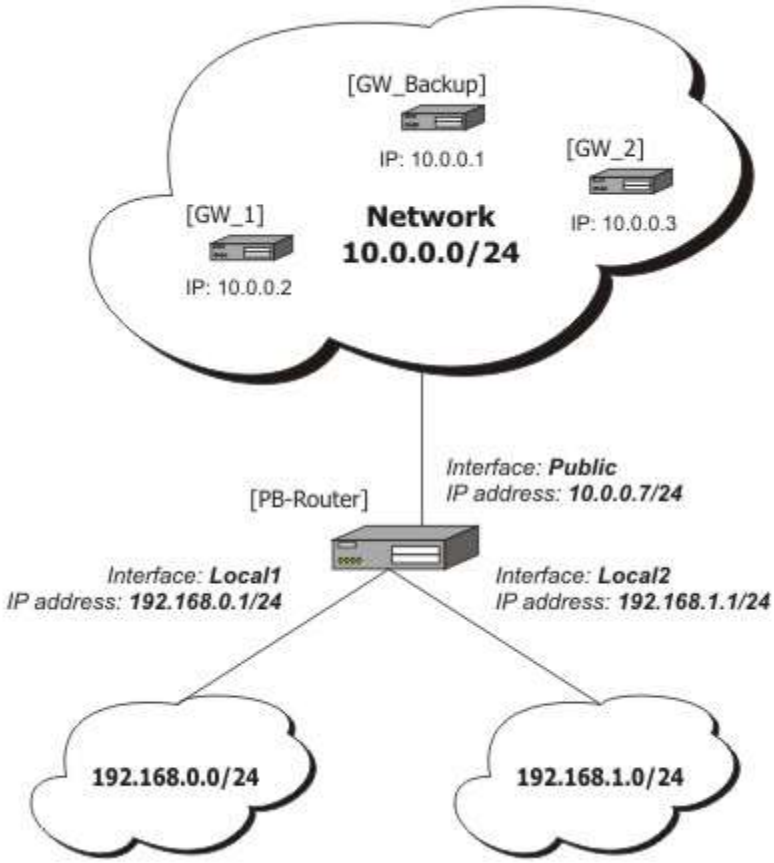
Add the default routes - one for ISP1 and 2 for ISP2 so we can get the ratio 1:3:

```
[admin@ECMP-Router] ip route> add gateway=10.1.0.1,10.1.1.1,10.1.1.1
[admin@ECMP-Router] ip route> print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf
 #     DST-ADDRESS        G GATEWAY         DISTANCE INTERFACE
 0 ADC 10.1.0.0/28                                   Public1
 1 ADC 10.1.1.0/28                                   Public2
 2 ADC 192.168.0.0/24                                Local
 3 A S 0.0.0.0/0          r 10.1.0.1                 Public1
                          r 10.1.1.1                 Public2
                          r 10.1.1.1                 Public2
[admin@ECMP-Router] ip route>
```

Standard Policy-Based Routing with Failover

This example will show how to route packets, using an administrator defined policy. The policy for this setup is the following: route packets from the network **192.168.0.0/24**, using gateway 10.0.0.1, and packets from network **192.168.1.0/24**, using gateway 10.0.0.2. If GW_1 does not respond to pings, use GW_Backup for network 192.168.0.0/24, if GW_2 does not respond to pings, use GW_Backup also for network 192.168.1.0/24 instead of GW_2.

The setup:



Configuration of the IP addresses:

```
[admin@PB-Router] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
 #   ADDRESS            NETWORK         BROADCAST       INTERFACE
 0   192.168.0.1/24     192.168.0.0     192.168.0.255   Local1
 1   192.168.1.1/24     192.168.1.0     192.168.1.255   Local2
 2   10.0.0.7/24        10.0.0.0        10.0.0.255      Public
[admin@PB-Router] ip address>
```

To achieve the described result, follow these configuration steps:

1. Mark packets from network 192.168.0.0/24 with a **new-routing-mark=net1**, and packets from network 192.168.1.0/24 with a **new-routing-mark=net2**:

```
[admin@PB-Router] ip firewall mangle> add src-address=192.168.0.0/24 \
\... action=mark-routing new-routing-mark=net1 chain=prerouting
[admin@PB-Router] ip firewall mangle> add src-address=192.168.1.0/24 \
\... action=mark-routing new-routing-mark=net2 chain=prerouting
[admin@PB-Router] ip firewall mangle> print
Flags: X - disabled, I - invalid, D - dynamic
 0   chain=prerouting src-address=192.168.0.0/24 action=mark-routing
     new-routing-mark=net1

 1   chain=prerouting src-address=192.168.1.0/24 action=mark-routing
     new-routing-mark=net2
```

```
[admin@PB-Router] ip firewall mangle>
```

13. Route packets from network 192.168.0.0/24 to gateway GW_1 (10.0.0.2), packets from network 192.168.1.0/24 to gateway GW_2 (10.0.0.3), using the according packet marks. If GW_1 or GW_2 fails (does not reply to pings), route the respective packets to GW_Main (10.0.0.1):

```
[admin@PB-Router] ip route> add gateway=10.0.0.2 routing-mark=net1 \
\... check-gateway=ping
[admin@PB-Router] ip route> add gateway=10.0.0.3 routing-mark=net2 \
\... check-gateway=ping
[admin@PB-Router] ip route> add gateway=10.0.0.1
[admin@PB-Router] ip route> print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf
 #     DST-ADDRESS        PREFSRC         G GATEWAY         DISTANCE
   INTERFACE
 0 ADC 10.0.0.0/24        10.0.0.7
   Public
 1 ADC 192.168.0.0/24     192.168.0.1
   Local1
 2 ADC 192.168.1.0/24     192.168.1.1
   Local2
 3 A S 0.0.0.0/0                          r 10.0.0.2
   Public
 4 A S 0.0.0.0/0                          r 10.0.0.3
   Public
 5 A S 0.0.0.0/0                          r 10.0.0.1
   Public
```

```
[admin@PB-Router] ip route>
```

HauteRouterOS Scripting Host

Document revision: 2.7 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

This manual provides introduction to RouterOS built-in powerful scripting language.

Scripting host provides a way to automate some router maintenance tasks by means of executing user-defined scripts bounded to some event occurence. A script consists of configuration commands and expressions (ICE - internal console expression). The configuration commands are standard RouterOS commands, e.g. `/ip firewall filter add chain=forward protocol=gre action=drop` that are described in the relevant manuals, while expressions are prefixed with **:** and are accessible from all submenus.

The events used to trigger script execution include, but are not limited to the System Scheduler, the Traffic Monitoring Tool, and the Netwatch Tool generated events.

Specifications

Packages required: ***system***
License required: *Level1*
Submenu level: ***/system script***
Standards and Technologies: None
Hardware usage: *Not significant*

Console Command Syntax

Description

Console commands are made of the following parts, listed in the order you type them in console:

- **prefix** - indicates whether the command is an ICE, like **:** in `:put` or that the command path starts from the root menu level, like **/** in

```
[admin@Hautespot] ip firewall mangle> /ping 10.0.0.1
```

- **path** - a relative path to the desired menu level, like **.. filter** in

```
[admin@Hautespot] ip firewall mangle> .. filter print
```

- **path_args** - this part is required to select some menu levels, where the actual path can vary across different user inputs, like **mylist** in

```
[admin@Hautespot] ip firewall mangle> /routeing prefix-list list mylist
```

- **action** - one of the actions available at the specified menu level, like **add** in

```
[admin@Hautespot] ip firewall mangle> /ip firewall filter add
chain=forward action=drop
```

- **unnamed parameter** - these are required by some actions and should be entered in fixed order after the action name, like in **10.0.0.1** in

```
[admin@Hautespot] ip firewall mangle> /ping 10.0.0.1
```

- **name[=value]** - a sequence of parameter names followed by respective values, if required, like **ssid=myssid** in

```
/interface wireless set wlan1 ssid=myssid
```

Notes

Variable substitution, command substitution and expressions are allowed only for **path_args** and **unnamed parameter** values. **prefix**, **path**, **action** and **name[=value]** pairs can be given only directly, as a word. Therefore, `:put (1 + 2)` is valid and `:("pu" . "t") 3` is not.

Example

The parts of internal console commands are futher explained in the following examples:

```
/ping 10.0.0.1 count=5
```

prefix	/
action	ping
unnamed parameter	10.0.0.1

name[=value]	count=5

```
.. ip firewall rule input
```

path	.. ip firewall rule
path_args	input

```
:for i from=1 to=10 do={:put $i}
```

prefix	:
action	for
unnamed parameter	i
pname[=value]	from=1 to=10 do={:put $i}

```
/interface monitor-traffic ether1,ether2,ipip1
```

prefix	/
path	interface
action	monitor-traffic
unnamed parameter	ether1,ether2,ipip1

Expression Grouping

Description

This feature provides an easy way to execute commands from within one command level, by enclosing them in braces '{ }'.

Notes

Subsequent script commands are executed from the same menu level as the entire script. Consider the following example:

```
[admin@Hautespot] ip route> /user {
{... /ip route
{... print}
Flags: X - disabled
 #   NAME                                                              GROUP
ADDRESS
 0   ;;; system default user
```

```
     admin                                                              full
0.0.0.0/0
 1   uuu                                                                full
0.0.0.0/0
[admin@Hautespot] ip route>
```

Although the current command level is changed to **/ip route**, it has no effect on next commands entered from prompt, therefore **print** command is still considered to be **/user print**.

Example

The example below demonstrates how to add two users to the **user** menu.

```
[admin@Hautespot] ip route> /user {
{... add name=x password=y group=write
{... add name=y password=z group=read
{... print}
Flags: X - disabled
 #   NAME                                                GROUP ADDRESS
 0   ;;; system default user
     admin                                               full  0.0.0.0/0
 1   x                                                   write 0.0.0.0/0
 2   y                                                   read  0.0.0.0/0
[admin@Hautespot] ip route>
```

Variables

Description

RouterOS scripting language suports two types of variables, which are global (system wide) and local (accessible only within the current script), respectively. A variable can be referenced by '$' (dollar) sign followed by the name of the variable with the exception of **set** and **unset** commands that take variable name without preceding dollar sign. Variable names should be composed of contain letters, digits and '-' character. A variable must be declared prior to using it in scripts. There are four types of declaration available:

- **global** - defined by **global** keyword, global variables can be accessed by all scripts and console logins on the same router. However, global variables are not kept across reboots.
- **local** - defined by **local** keyword, local variables are not shared with any other script, other instance of the same script or other console logins. The value of local variable value is lost when script finishes.
- **loop index variables** - defined within **for** and **foreach** statements, these variables are used only in **do** block of commands and are removed after command completes.

- **monitor variables** - some **monitor** commands that have **do** part can also introduce variables. You can obtain a list of available variables by placing **:environment print** statement inside the **do** block of commands.

You can assign a new value to variable using **set** action. It takes two unnamed parameters: the name of the variable and the new value of the variable. If a variable is no longer needed, it's name can be freed by **:unset** command. If you free local variable, it's value is lost. If you free global variable, it's value is still kept in router, it just becomes inaccessible from current script.

Notes

Loop variables "shadows" already introduced variables with the same name.

Example

```
[admin@Hautespot] ip route> /
[admin@Hautespot] > :global g1 "this is global variable"
[admin@Hautespot] > :put $g1
this is global variable
[admin@Hautespot] >
```

Command Substitution and Return Values

Description

Some console commands are most useful if their output can be feed to other commands as an argument value. In RouterOS console this is done by using the return values from commands. Return values are not displayed on the screen. To get the return value from a command, it should be enclosed in square brackets '[ ]'. Upon execution the return value of the the command will become the value of these brackets. This is called command substitution.

The commands that produce return values are, but not limited to: **find**, which returns a reference to a particular item, **ping**, which returns the number of sucessful pings, **time**, which returns the measured time value, **incr** and **decr**, which return the new value of a variable, and **add**, which returns the internal number of newly created item.

Example

Consider the usage of **find** command:

```
[admin@Hautespot] > /interface
[admin@Hautespot] interface> find type=ether
[admin@Hautespot] interface>
```

```
[admin@Hautespot] interface> :put [find type=ether]
*1,*2
[admin@Hautespot] interface>
```

This way you can see internal console numbers of items. Naturally, you can use them as arguments in other commands:

```
[admin@Hautespot] interface> enable [find type=ether]
[admin@Hautespot] interface>
```

Operators

Description

RouterOS console can do simple calculations with numbers, time values, IP addresses, strings and lists. To get result from an expression with operators, enclose it in parentheses '(' and ')'. The expression result serves as a return value for the parentheses.

Command Description

- - unary minus. Inverts given number value.
- - binary minus. Substracts two numbers, two time values, two IP addresses or an IP address and a number
! - logical **NOT**. Unary operator, which inverts given boolean value
/ - division. Binary operator. Divides one number by another (gives number) or a time value by a number (gives time value).
. - concatenation. Binary operator, concatenates two string or append one list to another or appends an element to a list.
^ - bitwise **XOR**. The argumens and the result are both IP addresses
~ - bit inversion. Unary operator, which inverts bits in IP address
***** - multiplication. Binary operator, which can multiply two numbers or a time value by a number.
& - bitwise **AND** The argumens and the result are both IP addresses
&& - logical **AND**. Binary operator. The argumens and the result are both logical values
+ - binary plus. Adds two numbers, two time values or a number and an IP address.
< - less. Binary operator which compares two numbers, two time values or two IP addresses. Returns boolean value
<< - left shift. Binary operator, which shifts IP address by a given amount of bits. The first argument is an IP address, the second is an integer and the result is an IP address.
<= - less or equal. Binary operator which compares two numbers, two time values or two IP addresses. Returns boolean value
> - greater. Binary operator which compares two numbers, two time values or two IP addresses. Returns boolean value

>= - greater or equal. Binary operator which compares two numbers, two time values or two IP addresses. Returns boolean value
>> - right shift. Binary operator, which shifts IP address by a given amount of bits. The first argument is an IP address, the second is an integer and the result is an IP address.
| - bitwise **OR**. The argumens and the result are both IP addresses
|| - logical **OR**. Binary operator. The argumens and the result are both logical values

Notes

When comparing two arrays note, that two arrays are equal only if their respective elements are equal.

Example

Operator priority and evaluation order

```
[admin@Hautespot] ip firewall rule forward> :put (10+1-6*2=11-12=2+(-3)=-1)
false
[admin@Hautespot] ip firewall rule forward> :put (10+1-6*2=11-12=(2+(-3)=-1))
true
[admin@Hautespot] ip firewall rule forward
```

logical NOT

```
[admin@Hautespot] interface> :put (!true)
false
[admin@Hautespot] interface> :put (!(2>3))
true
[admin@Hautespot] interface>
```

unary minus

```
[admin@Hautespot] interface> :put (-1<0)
true
[admin@Hautespot] >
1
```

bit inversion

```
[admin@Hautespot] interface> :put (~255.255.0.0)
0.0.255.255
[admin@Hautespot] interface>
```

sum

```
[admin@Hautespot] interface> :put (3ms + 5s)
00:00:05.003
[admin@Hautespot] interface> :put (10.0.0.15 + 0.0.10.0)
cannot add ip address to ip address
[admin@Hautespot] interface> :put (10.0.0.15 + 10)
10.0.0.25
[admin@Hautespot] interface>
```

subtraction

```
[admin@Hautespot] interface> :put (15 - 10)
5
[admin@Hautespot] interface> :put (10.0.0.15 - 10.0.0.3)
12
[admin@Hautespot] interface> :put (10.0.0.15 - 12)
10.0.0.3
[admin@Hautespot] interface> :put (15h - 2s)
14:59:58
[admin@Hautespot] interface>
```

multiplication

```
[admin@Hautespot] interface> :put (12s * 4)
00:00:48
[admin@Hautespot] interface> :put (-5 * -2)
10
[admin@Hautespot] interface>
```

division

```
[admin@Hautespot] interface> :put (10s / 3)
00:00:03.333
[admin@Hautespot] interface> :put (5 / 2)
2
[admin@Hautespot] interface>
[admin@Hautespot] > :put (0:0.10 / 3)
00:00:02
[admin@Hautespot] >
```

comparison

```
[admin@Hautespot] interface> :put (10.0.2.3<=2.0.3.10)
false
[admin@Hautespot] interface> :put (100000s>27h)
true
[admin@Hautespot] interface> :put (60s,1d!=1m,3600s)
true
[admin@Hautespot] interface> :put (bridge=routing)
false
```

```
[admin@Hautespot] interface> :put (yes=false)
false
[admin@Hautespot] interface> :put (true=aye)
false
[admin@Hautespot] interface>
```

logical AND, logical OR

```
[admin@Hautespot] interface> :put ((yes && yes) || (yes && no))
true
[admin@Hautespot] interface> :put ((no || no) && (no || yes))
false
[admin@Hautespot] interface>
```

bitwise AND, bitwise OR, bitwise XOR

```
[admin@Hautespot] interface> :put (10.16.0.134 & ~255.255.255.0)
0.0.0.134
[admin@Hautespot] interface>
```

shift operators

```
[admin@Hautespot] interface> :put (~((0.0.0.1 << 7) - 1))
255.255.255.128
[admin@Hautespot] interface>
```

Concatenation

```
[admin@Hautespot] interface> :put (1 . 3)
13
[admin@Hautespot] interface> :put (1,2 . 3)
1,2,3
[admin@Hautespot] interface> :put (1 . 3,4)
13,4
[admin@Hautespot] interface> :put (1,2 . 3,4)
1,2,3,4
[admin@Hautespot] interface> :put ((1 . 3) + 1)
14
[admin@Hautespot] interface>
```

Data types

Description

The RouterOS console differentiates between several data types, which are string, boolean, number, time interval, IP address, internal number and list. The console tries to convert any value to the most specific type first, backing if it fails. The order in which the console attempts to convert an entered value is presented below:

- list
- internal number
- number
- IP address
- time
- boolean
- string

Internal scripting language supplies special functions to explicitly control type conversion. The **toarray**, **tobool**, **toid**, **toip**, **tonum**, **tostr** and **totime** functions convert a value accordingly to **list**, **boolean**, **internal number**, **IP address**, **number**, **string** or **time**.

The number type is internally represented as 64 bit signed integer, so the value a number type variable can take is in range from -9223372036854775808 to 9223372036854775807. It is possible to input number value in hexadecimal form, by prefixing it with **0x**, e.g.:

```
[admin@Hautespot] > :global MyVar 0x10
[admin@Hautespot] > :put $MyVar
16
[admin@Hautespot] >
```

Lists are treated as comma separated sequence of values. Putting whitespaces around commas is not recommended, because it might confuse console about words' boundaries.

Boolean values can be either **true** or **false**. Console also accepts **yes** for **true**, and **no** for **false**.

Internal numbers are preceided ***** sign.

Time intervals can be entered either using HH:MM:SS.MS notation, e.g.:

```
[admin@Hautespot] > :put 01:12:1.01
01:12:01.010
[admin@Hautespot] >
```

or as sequence of numbers, optionally followed by letters specifying the units of time measure (**d** dor days, **h** for hours, **m** for minutes, **s** for seconds and **ms** for milliseconds), e.g.:

```
[admin@Hautespot] > :put 2d11h12
2d11:00:12
[admin@Hautespot] >
```

As can bee seen, time values with omitted unit specificators are treated as expressed in seconds.

Possible aliases for time units:

- **d, day, days** - one day, or 24 hours
- **h, hour, hours** - one hour
- **m, min** - one minute
- **s** - one second
- **ms** - one millisecond, *id est* 0.001 second

The console also accepts time values with decimal point:

```
[admin@Hautespot] > :put 0.1day1.2s
02:24:01.200
[admin@Hautespot] >
```

Command Reference

Description

RouterOS has a number of built-in console commands and expressions (ICE) that do not depend on the current menu level. These commands do not change configuration directly, but they are useful for automating various maintenance tasks. The full ICE list can be accessed by typing '?' after the ':' prefix (therefore it can be safely assumed that all ICE have ':' prefix), for example:

```
[admin@Hautespot] > :
beep         execute  global  list     pick     time     toip    typeof
delay        find     if      local    put      toarray  tonum   while
do           for      led     log      resolve  tobool   tostr
environment  foreach  len     nothing  set      toid     totime
[admin@Hautespot] >
```

Command Description

beep - forces the built-in PC beeper to produce a signal for **length** seconds at **frequency** Hz.
Input Parameters
frequency (*integer*; default: **1000**) - signal frequency measured in Hz
length (*time*; default: **100ms**) - signal length

```
[admin@Hautespot] > :beep length=2s frequency=10000

[admin@Hautespot] >
```

delay - does nothing for a given amount of time.

Input Parameters

delay-time (*time*) - amount of time to wait

- **omitted** - delay forever

do - executes commands repeatedly until given conditions are met. If no parameters are given, **do** just executes its payload once, which does not make much use. If a logical condition is specified for the **while** parameter, it will be evaluated after executing commands, and in case it is *true*, **do** statement is executed again and again until *false*. The **if** parameter, if present, is evaluated only once before doing anything else, and if it is *false* then no action is taken

Input Parameters

unnamed (*text*) - actions to execute repeatedly

while (yes | no) - condition, which is evaluated each time after the execution of enclosed statements

if (yes | no) - condition, which is evaluated once before the execution of enclosed statements

```
[admin@Hautespot] > {:global i 10; :do {:put $i; :set i ($i - 1);} \
\... while (($i < 11) && ($i > 0)); :unset i;}
10
9
8
7
6
5
4
3
2
1
[admin@Hautespot] >
```

environment print - prints information about variables that are currently initialised. All global variables in the system are listed under the heading **Global Variables**. All variables that are introduced in the current script (variables introduced by **:local** or created by **:for** or **:foreach** statements) are listed under the heading **Local Variables.**

Creating variables and displaying a list of them

```
[admin@Hautespot] > :local A "This is a local variable"
[admin@Hautespot] > :global B "This is a global one"
[admin@Hautespot] > :environment print
Global Variables
B=This is a global one
Local Variables
A=This is a local variable
[admin@Hautespot] >
```

find - searches for substring inside a string or for an element with particular value inside an array, depending on argument types and returns position at which the value is found. The elements in list and characters in string are numbered from 0 upwards

Input Parameters

unnamed(*text* | *list*) - the string or value list the search will be performed in

unnamed(*text*) - value to be searched for
unnamed(*integer*) - position after which the search is started

```
[admin@Hautespot] interface pppoe-server> :put [:find
"13sdf1sdfss1sfsdf324333"  ]
0
[admin@Hautespot] interface pppoe-server> :put [:find
"13sdf1sdfss1sfsdf324333" 3  ]
1
[admin@Hautespot] interface pppoe-server> :put [:find
"13sdf1sdfss1sfsdf324333" 3  3]
17
[admin@Hautespot] interface pppoe-server> :put [:find
"1,1,1,2,3,3,4,5,6,7,8,9,0,1,2,3" 3 ]
4
[admin@Hautespot] interface pppoe-server> :put [:find
"1,1,1,2,3,3,4,5,6,7,8,9,0,1,2,3" 3 3]
4
[admin@Hautespot] interface pppoe-server> :put [:find
"1,1,1,2,3,3,4,5,6,7,8,9,0,1,2,3" 3 4]
5
[admin@Hautespot] interface pppoe-server> :put [:find
"1,1,1,2,3,3,4,5,6,7,8,9,0,1,2,3" 3 5]
15
[admin@Hautespot]
```

for - executes supplied commands over a given number of iterations, which is explicity set through **from** and **to** parameters

Input Parameters

unnamed (*name*) - the name of the loop counter variable
from (*integer*) - start value of the loop counter variable
to (*integer*) - end value of the loop counter variable
step (*integer*; default: **1**) - increment value. Depending on the loop counter variable start and end values, **step** parameter can be treated also as decrement
do (*text*) - contains the command to be executed repeatedly

```
[admin@Hautespot] > :for i from=1 to=100 step=37  do={:put ($i . " - " .
1000/$i)}
1 - 1000
38 - 26
75 - 13
[admin@Hautespot] >
```

foreach - executes supplied commands for each element in list

Input Parameters

unnamed (*name*) - the name of the loop counter variable
in (*list*) - list of values over which to iterate
do (*text*) - contains the command to be executed repeatedly

Printing a list of available interfaces with their respective IP addresses

```
:foreach i in=[/interface find type=ether ] \
\... do={:put ("+--" . [/interface get $i name]); \
\... :foreach j in=[/ip address find interface=$i]
\... do={:put ("|  `--" . [/ip address get $j address])}}
+--ether1
|  `--1.1.1.3/24
|  `--192.168.50.1/24
|  `--10.0.0.2/24
+--ether2
|  `--10.10.0.2/24
```

```
[admin@Hautespot] >
```

global - declares global variable

Input Parameters

unnamed(*name*) - name of the variable

unnamed(*text*) - value, which should be assigned to the variable

```
[admin@Hautespot] > :global MyString "This is a string"
[admin@Hautespot] > :global IPAddr 10.0.0.1
[admin@Hautespot] > :global time   0:10
[admin@Hautespot] > :environment print
Global Variables
IPAddr=10.0.0.1
time=00:10:00
MyString=This is a string
Local Variables
[admin@Hautespot] >
```

if - conditional statement. If a given logical condition evaluates to true then the **do** block of commands is executed. Otherwice an optional **else** block is executed.

Input Parameters

unnamed(yes | no) - logical condition, which is evaluated once before the execution of enclosed statements

do(*text*) - this block of commands is executed if the logical condition evaluates to *true*

else(*text*) - this block of commands is executed if the logical condition evaluates to *false*

Check if the firewall has any rules added

```
[admin@Hautespot] > :if ([:len [/ip firewall filter find]] > 0) do={:put true}
else={:put false}
true
[admin@Hautespot] >
```

Check whether the gateway is reachable. In this example, the IP address of the gateway is **10.0.0.254**

```
[admin@Hautespot] > :if ([/ping 10.0.0.254 count=1] = 0) do {:put "gateway
unreachable"}
10.0.0.254 ping timeout
1 packets transmitted, 0 packets received, 100% packet loss
gateway unreachable
[admin@Hautespot] >
```

led - allows to control system built-in LEDs (Light Emitting Diodes). This command is available only on HauteWRAP platforms whith the **HauteWRAP** or **rb500** packages installed, depending on the HauteWRAP model. The number of available LEDs varies between different HauteWRAP models

Input Parameters

led1(yes | no) - controls first LED

led2(yes | no) - controls second LED

led3(yes | no) - controls third LED

led4(yes | no) - controls fourth LED

length(*time*) - specifies the length of the action

- **omitted** - altar LED state forever

Switch on LEDs 2 and 3 for 5 seconds

```
[admin@Hautespot] > :led led2=yes led3=yes length=5s
```

len - returns the number of characters in string or the number of elements in list depending on the type of the argument

Input Parameters

unnamed(*name*) - string or list the length of which should be returned

```
[admin@Hautespot] > :put [:len gvejimezyfopmekun]
17
[admin@Hautespot] > :put [:len gve,jim,ezy,fop,mek,un]
6
[admin@Hautespot] >
```

list - displays a list of all available console commands that match given search key(s)

Input Parameters

unnamed(*text*) - first search key
unnamed(*text*) - second search key
unnamed(*text*) - third search key

Display console commands that have **hotspot**, **add** and **user** parts in the command's name and path

```
[admin@Hautespot] > :list user hotspot "add "
List of console commands under "/" matching "user" and "hotspot" and "add ":

ip hotspot profile add name= hotspot-address= dns-name= \
\... html-directory= rate-limit= http-proxy= smtp-server= \
\... login-by= http-cookie-lifetime= ssl-certificate= split-user-domain= \
\... use-radius= radius-accounting= radius-interim-update= copy-from=
ip hotspot user add server= name= password= address= mac-address= \
\... profile= routes= limit-uptime= limit-bytes-in= limit-bytes-out= \
\... copy-from= comment= disabled=
ip hotspot user profile add name= address-pool= session-timeout= \
\... idle-timeout= keepalive-timeout= status-autorefresh= \
\... shared-users= rate-limit= incoming-filter= outgoing-filter= \
\... incoming-mark= outgoing-mark= open-status-page= on-login= on-logout=
copy-from=
[admin@Hautespot] >
```

local - declares local variable

Input Parameters

unnamed(*name*) - name of the variable
unnamed(*text*) - value, which should be assigned to the variable

```
[admin@Hautespot] > :local MyString "This is a string"
[admin@Hautespot] > :local IPAddr 10.0.0.1
[admin@Hautespot] > :local time   0:10
[admin@Hautespot] > :environment print
Global Variables
Local Variables
IPAddr=10.0.0.1
time=00:10:00
MyString=This is a string
[admin@Hautespot] >
```

log - adds a message specified by **message** parameter to the system logs.

Input Parameters

unnamed(*name*) - name of the logging facility to send message to
unnamed(*text*) - the text of the message to be logged

Send message to **info** log

```
[admin@Hautespot] > :log info "Very Good thing happened. We have received our
first packet!"
[admin@Hautespot] > /log print follow
...
19:57:46 script,info Very Good thing happened. We have received our first
packet!
...
```

nothing - has no action, and returns value of type "nothing". In conditions nothing behaves as "false"

Pick a symbol that does not exist from a string

```
[admin@Hautespot] > :local string qwerty
[admin@Hautespot] > :if ([:pick $string 10]=[:nothing]) do={
{... :put "pick and nothing commands return the same value"}
pick and nothing commands return the same value
[admin@Hautespot] >
```

pick - returns a range of elements or a substring depending on the type of input value

Input Parameters

unnamed(*text* | *list*) - the string or value list from which a substring or a subrange should be returned
unnamed(*integer*) - start position of substring or subrange
unnamed(*integer*) - end position for substring or subrange

```
[admin@Hautespot] > :set a 1,2,3,4,5,6,7,8
[admin@Hautespot] > :put [:len $a]
8
[admin@Hautespot] > :put [:pick $a]
1
[admin@Hautespot] > :put [:pick $a 0 4]
1,2,3,4
[admin@Hautespot] > :put [:pick $a 2 4]
3,4
[admin@Hautespot] > :put [:pick $a 2]
3
[admin@Hautespot] > :put [:pick $a 5 1000000]
6,7,8
[admin@Hautespot] > :set a abcdefghij
[admin@Hautespot] > :put [:len $a]
10
[admin@Hautespot] > :put [:pick $a]
a
[admin@Hautespot] > :put [:pick $a 0 4]
abcd
[admin@Hautespot] > :put [:pick $a 2 4]
cd
[admin@Hautespot] > :put [:pick $a 2]
c
[admin@Hautespot] > :put [:pick $a 5 1000000]
fghij
```

put - echoes supplied argument to the console

Input Parameters

unnamed(*text*) - the text to be echoed to the console

Display the MTU of **ether1** interface

```
[admin@Hautespot] > :put [/interface get ether1 mtu]
```

```
1500
[admin@Hautespot] >
```

resolve - returns IP address of the host resolved from the DNS name. The DNS settings should be configured on the router (**/ip dns** submenu) prior to using this command.

Input Parameters

unnamed(*text*) - domain name to be resolved into an IP address

DNS configuration and **resolve** command example

```
[admin@Hautespot] ip route> /ip dns set primary-dns=159.148.60.2
[admin@Hautespot] ip route> :put [:resolve www.example.com]
192.0.34.166
```

set - assigns new value to a variable

Input Parameters

unnamed(*name*) - the name of the variable

unnamed(*text*) - the new value of the variable

Measuring time needed to resolve www.example.com

```
[admin@Hautespot] > :put [:time [:resolve www.example.com ]]
00:00:00.006
[admin@Hautespot] >
```

time - measures the amount of time needed to execute given console commands

Input Parameters

unnamed(*text*) - the console commands to measure execution time of

Measuring time needed to resolve www.example.com

```
[admin@Hautespot] > :put [:time [:resolve www.example.com ]]
00:00:00.006
[admin@Hautespot] >
```

while - executes given console commands repeatedly while the logical conditions is *true*

Input Parameters

unnamed(yes | no) - condition, which is evaluated each time before the execution of enclosed statements

do(*text*) - console commands that should be executed repeatedly

```
[admin@Hautespot] > :set i 0; :while ($i < 10) do={:put $i; :set i ($i + 1)};
0
1
2
3
4
5
6
7
8
9
[admin@Hautespot] >
```

Special Commands

Description

Monitor

It is possible to access values that are shown by most **monitor** actions from scripts. A **monitor** command that has a **do** parameter can be supplied either script name (see **/system scripts**), or console commands to execute.

Get

Most **print** commands produce values that are accessible from scripts. Such **print** commands have a corresponding **get** command on the same menu level. The **get** command accepts one parameter when working with regular values or two parameters when working with lists.

Notes

Monitor command with **do** argument can also be called directly from scripts. It will not print anything then, just execute the given script.

The names of the properties that can be accessed by **get** are the same as shown by **print** command, plus names of item flags (like the disabled in the example below). You can use [T ab] key completions to see what properties any particular **get** action can return.

Example

In the example below **monitor** action will execute given script each time it prints stats on the screen, and it will assign all printed values to local variables with the same name:

```
[admin@Hautespot] interface> monitor-traffic ether2 once do={:environment
print}
    received-packets-per-second: 0
       received-bits-per-second: 0bps
        sent-packets-per-second: 0
           sent-bits-per-second: 0bps

Global Variables
i=1
Local Variables
sent-bits-per-second=0
received-packets-per-second=0
received-bits-per-second=0
sent-packets-per-second=0
[admin@Hautespot] interface>
```

Additional Features

Description

To include comment in the console script prefix it with '#'. In a line of script that starts with '#' all characters until the newline character are ignored.

To put multiple commands on a single line separate them with ';'. Console treats ';' as the end of line in scripts.

Any of the **{}[]"'\$** characters should be escaped in a regular string with '\' character. Console takes any character following '\' literally, without assigning any special meaning to it, except for such cases:

```
\a     bell (alarm), character code 7
\b     backspace, character code 8
\f     form feed, character code 12
\n     newline, character code 10
\r     carriage return, character code 13
\t     tabulation, character code 9
\v     vertical tabulation, character code 11
\_     space, character code 32
```

Note that '\', followed by any amount of whitespace characters (spaces, newlines, carriage returns, tabulations), followed by newline is treated as a single whitespace, except inside quotes, where it is treated as nothing. This is used by console to break up long lines in scripts generated by export commands.

Script Repository

Submenu level: ***/system script***

Description

All scripts are stored in the **/system script** menu along with some service information such as script name, script owner, number of times the script was executed and permissions for particular script.

In RouterOS, a script may be automatically started in three different ways:

- via the scheduler
- on event occurence - for example, the netwatch tool generates an event if a network host it is configured to monitor becomes unaccessible
- by another script

It is also possible to start a script manually via **/system script run** command.

Property Description

last-started (*time*) - date and time when the script has been last invoked. The argument is shown only if the **run-count**!=**0**.
owner (*name*; default: **admin**) - the name of the user who created the script
policy (*multiple choice:* ftp | local | policy | read | reboot | ssh | telnet | test | web | write; default: **reboot,read,write,policy,test**) - the list of the policies applicable:
ftp - user can log on remotely via ftp and send and retrieve files from the router
local - user can log on locally via console
policy - manage user policies, add and remove user
read - user can retrieve the configuration
reboot - user can reboot the router
ssh - user can log on remotely via secure shell
telnet - user can log on remotely via telnet
test - user can run ping, traceroute, bandwidth test
web - user can log on remotely via http
write - user can retrieve and change the configuration
run-count (*integer*; default: **0**) - script usage counter. This counter is incremented each time the script is executed. The counter will reset after reboot.
source (*text*; default: **""**) - the script source code itself

Command Description

run (*name*) - executes a given script
Input Parameters
unnamed(*name*) - the name of the script to execute

Notes

You cannot do more in scripts than you are allowed to do by your current user rights, that is, you cannot use disabled policies. For example, if there is a policy group in **/user group** which allows you **ssh,local,telnet,read,write,policy,test,web** and this group is assigned to your user name, then you cannot make a script that reboots the router.

Example

The following example is a script for writing message "Hello World!" to the **info** log:

```
[admin@Hautespot] system script> add name="log-test" source={:log info "Hello
World!"}
[admin@Hautespot] system script> run log-test

[admin@Hautespot] system script> print
 0 name="log-test" owner="admin"
policy=ftp,reboot,read,write,policy,test,HSConfig,password last-
started=mar/20/2001 22:51:41
   run-count=1 source=:log info "Hello World!"
[admin@Hautespot] system script>
```

Task Management

Submenu level: ***/system script job***

Description

This facility is used to manage the active or scheduled tasks.

Property Description

name (*read-only: name*) - the name of the script to be referenced when invoking it
owner (*text*) - the name of the user who created the script
source (*read-only: text*) - the script source code itself

Example

```
[admin@Hautespot] system script> job print
  # SCRIPT  OWNER                   STARTED
  0 DelayeD admin                   dec/27/2003 11:17:33

[admin@Hautespot] system script>
```

You can cancel execution of a script by removing it from the job list

```
[admin@Hautespot] system script> job remove 0
[admin@Hautespot] system script> job print

[admin@Hautespot] system script>
```

Script Editor

Command name: ***/system script edit***

Description

RouterOS console has a simple full-screen editor for scripts with support for multiline script writing.

Keyboard Shortcuts

- **Delete** - deletes character at cursor position
- **Ctrl+h, backspase** - deletes character before cursor. Unindents line
- **Tab** - indents line
- **Ctrl+b, LeftArrow** - moves cursor left
- **Ctrl+f, RightArrow** - moves cursor right
- **Ctrl+p, UpArrow** - moves cursor up
- **Ctrl+n, DownArrow** - moves cursor down

- **Ctrl+a, Home** - moves cursor to the beginning of line or script
- **Ctrl+e, End** - moves cursor to the end of line or script
- **Ctrl+y** - inserts contents of buffer at cursor position
- **Ctrl+k** - deletes characters from cursor position to the end of line
- **Ctrl+u** - undoes last action
- **Ctrl+o** - exits editor accepting changes
- **Ctrl+x** - exits editor discarding changes

Command Description

edit (*name*) - opens the script specified by the **name** argument in full-screen editor

Notes

All characters that are deleted by **backspace, delete** or **Ctrl+k** keys are accumulated in the buffer. Pressing any other key finishes adding to this buffer (**Ctrl+y** can paste it's contents), and the next delete operation will replace it's contents. Undo doesn't change contents of cut buffer.

Script editor works only on VT102 compatible terminals (terminal names "vt102", "linux", "xterm", "rxvt" are recognized as VT102 at the moment). Delete, backspace and cursor keys might not work with all terminal programs, use 'Ctrl' alternatives in such cases.

HauteRouterOS Serial Console and Terminal

Document revision: 2.1 (Tuesday, October 30, 2007)

Applies to: HauteRouterOS V3.0

General Information

Summary

The Serial Console and Terminal are tools, used to communicate with devices and other systems that are interconnected via serial port. The serial terminal may be used to monitor and configure many devices - including modems, network devices (including HauteSpot routers), and any device that can be connected to a serial (asynchronous) port.

Specifications

Packages required: ***system***
License required: *Level1*
Submenu level: ***/system***, ***/system console***, ***/system serial-terminal***
Standards and Technologies: RS-232
Hardware usage: *Not significant*

Related Documents

- Software Package Management

Description

The Serial Console (managed side) feature allows configuring one serial port of the HauteSpot router for access to the router's Terminal Console over the serial port. A special null-modem cable is required to connect the router's serial port with the workstation's or laptop's serial (COM) port. A terminal emulation program, e.g., HyperTerminal, should be run on the workstation. You can also use HauteRouterOS to connect to an another Serial Console (for example, on a Cisco router).

Several customers have described situations where the Serial Terminal (managing side) feature would be useful:

- in a mountaintop where a HauteSpot wireless installation sits next to equipment (including switches and Cisco routers) that can not be managed in-band (by telnet through an IP network)
- monitoring weather-reporting equipment through a serial-console
- connection to a high-speed microwave modem that needed to be monitored and managed by a serial-console connection

With the serial-terminal feature of the HauteSpot, up to 132 (and, maybe, even more) devices can be monitored and controlled

Serial Console Configuration

Description

A special null-modem cable should be used for connecting to the serial console. The Serial Console cabling diagram for DB9 connectors is as follows:

Router Side (DB9f)	Signal	Direction	Side (DB9f)
1, 6	CD, DSR	IN	4
2	RxD	IN	3
3	TxD	OUT	2

4	DTR	OUT	1, 6
5	GND	-	5
7	RTS	OUT	8
8	CTS	IN	7

Configuring Console

Submenu level: ***/system console***

Property Description

enabled (yes | no; default: **no**) - whether serial console is enabled or not
free (*read-only: text*) - console is ready for use
port (*name*; default: **serial0**) - which port should the serial terminal listen to
term (*text*) - name for the terminal
used (*read-only: text*) - console is in use
vcno (*read-only: integer*) - number of virtual console - [Alt]+[F1] represents '1', [Alt]+[F2] - '2', etc.
wedged (*read-only: text*) - console is currently not available

Example

To enable Serial Console with terminal name **MyConsole**:

```
[admin@HauteSpot] system console> set 0 disabled=no term=MyConsole
[admin@HauteSpot] system console> print
Flags: X - disabled, W - wedged, U - used, F - free
 #   PORT    VCNO       TERM
 0 F serial0            MyConsole
 1 W           1          linux
 2 W           2          linux
 3 W           3          linux
 4 W           4          linux
 5 W           5          linux
 6 W           6          linux
 7 W           7          linux
 8 W           8          linux
[admin@HauteSpot] system console>
```

To check if the port is available or used (parameter **used-by**):

```
[admin@HauteSpot] system serial-console> /port print detail
  0 name=serial0 used-by=Serial Console baud-rate=9600 data-bits=8 parity=none
    stop-bits=1 flow-control=none
```

```
  1 name=serial1 used-by="" baud-rate=9600 data-bits=8 parity=none stop-bits=1
    flow-control=none

[admin@HauteSpot] system serial-console>
```

Using Serial Terminal

Command name: ***/system serial-terminal***

Description

The command is used to communicate with devices and other systems that are connected to router via serial port.

All keyboard input is forwarded to the serial port and all data from the port is output to the connected device. After exiting with [Ctrl]+[Q], the control signals of the port are lowered. The speed and other parameters of serial port may be configured in the **/port** directory of router console. No terminal translation on printed data is performed. It is possible to get the terminal in an unusable state by outputting sequences of inappropriate control characters or random data. Do not connect to devices at an incorrect speed and avoid dumping binary data.

Property Description

port (*name*) - port name to use

Notes

[Ctrl]+[Q] and [Ctrl]+[X] have special meaning and are used to provide a possibility of exiting from nested serial-terminal sessions:

To send [Ctrl]+[X] to to serial port, press [Ctrl]+[X] [Ctrl]+[X]

To send [Ctrl]+[Q] to to serial port, press [Ctrl]+[X] [Ctrl]+[Q]

Example

To connect to a device connected to the **serial1** port:

```
[admin@HauteSpot] system> serial-terminal serial1

[Type Ctrl-Q to return to console]
[Ctrl-X is the prefix key]
```

Console Screen

Submenu level: ***/system console screen***

Description

This facility is created to change line number per screen if you have a monitor connected to router.

Property Description

line-count (25 | 40 | 50) - number of lines on monitor

Notes

This parameter is applied only to a monitor, connected to the router.

Example

To set monitor's resolution from 80x25 to 80x40:

```
[admin@HauteSpot] system console screen> set line-count=40
[admin@HauteSpot] system console screen> print
    line-count: 40
[admin@HauteSpot] system console screen>
```

Setting Up HauteRouteOS and FreeRADIUS

First install FreeRadius and MySQL from RPM, apt, yum or source.

Once FreeRADIUS is installed, we now need to configure it. This step will detail how to setup the server for use with the local Unix user accounts for the machine that FreeRADIUS is installed on. If you would like to immediately setup the server for use with the MySQL database proceed to the next step but I highly recommend you do this step first to verify the RADIUS install works properly. Step 4 also builds on this step. First we are going to need some tool to test the installation of the RADIUS server with, I prefer NTRadPing you can download it from MasterSoft's website at: http://www.mastersoft-group.com/download/ NTRadPing provides a nice simple testing interface for MS Windows computers.

First we need to authorize access to the RADIUS server to certain computers:

```
cd /etc/raddb
```

```
vi clients
```

Add the IP address of the HR-WRAP and the IP address of the windows computer you have NTRadPing installed on and pick a secret key for each.

Example:

```
# Client Name Key
#---------------- ----------
192.168.1.100 testkey
ramona.lb.ru testkey
```

Modern FreeRADIUS distributions include the HauteSpot dictionary, but you need to check to make sure if it is there. Look in /usr/local/share/freeradius to see if dictionary.HauteSpot is present, if not we now need to 'install' it:

```
cd /usr/local/share/freeradius
wget
http://www.HauteSpot.com/Documentation/manual_2.9/dictionary.HauteSpot
```

Then edit the dictionary file

```
vi dictionary
```

After the last $INCLUDE statement add the following line:

```
$INCLUDE dictionary.HauteSpot
```

naslist

Add the same IP addresses for your test computer and HR-WRAP box into this file and select the type of NAS. Example:

```
# NAS Name Short Name Type
#---------------- ---------- ----
localhost local portslave
192.168.1.100 test portslave
Hrwrap.hautespot.net HauteSpot
```

radiusd.conf

Find the Unix section of the file and ensure that the lines

```
passwd = /etc/passwd
shadow = /etc/shadow
group = /etc/group
```

are NOT commented out (ie. do not have a # sign in front of them).

Congratulations!!! You now have a fully functional RADIUS server that will uses the local Unix accounts as its authentication base.

Let's test it out

Start the RADIUS server in debug mode(-x) by typing:

```
radiusd –x
```

- Note: You must be logged in as root (su or real login) to start the server, otherwise you will get a "command not found" error.

If you receive no error messages you have configured the server properly, now on the MS Windows machine open NTRadPing. Enter the following:

```
RADIUS Server: {ip address of your radius server}
Port: 1812
RADIUS Secret Key: {the key you specified in the clients file}
User-Name: root
Password: {root password for the machine on which freeRADIUS is
installed}
```

- You should get an "Access-Accept" message.
- Now change the password to something incorrect, ensure you get an "Access-Reject" message
- Now correct the password, change the port to 1813, and change the Request Type to: "Accounting Start". Ensure you get an "Accounting-Response" message.
- Finally change the Request Type to: "Accounting Stop". Ensure you get an "Accounting-Response" message.

MySQL Database

If the above tests came back in good order we can go ahead and setup FreeRadius to use the MySQL database.

Stop the RADIUS server by pressing CTRL-C

First of all we will need to setup a database schema for the RADIUS server to use, fortunately FreeRADIUS ships with a SQL script to create the database for us. In this step when I refer to root and the respective root password, I am referring to the root user for the MySQL database, not root of the entire system.

```
cd /usr/bin
mysql –uroot –p{root password}
```

You are now in the MySQL shell, all SQL commands end with a semicolon (Wink

We need to create a database before we can create tables within it:

```
CREATE DATABASE radius;
```

This creates an empty database named “radius”

```
exit
```

Now execute the script included with FreeRADIUS to create the database tables:

```
mysql -uroot -pccsccs radius <
/software/freeradius/freeradiusxxx/src/modules
/rlm_sql/drivers/rlm_sql_mysql/db_mysql.sql

mysql –uroot –p{root password}
USE radius;
SHOW TABLES;
```

You should see 6 different tables in the radius database.

Let’s insert a test user while we are at it:

```
INSERT INTO usergroup (UserName, GroupName) {press enter, it will bring
you to a new line}
VALUES (“radiustest”, “testgroup”);
```

Verify the user was added by viewing the rows of the usergroup table:

```
SELECT * FROM usergroup;
```

We still need to set more attributes for the user:

```
INSERT INTO radcheck (UserName, Attribute, Value)
VLALUES (“radiustest”, “Password”, “testpassword”);
Verify it is in there:
SELECT * FROM radcheck;
```

We will configure a quick test response here as well (not to be used in the final implementation, you will have to configure that depending on what you want, this will work however):

```
INSERT INTO radgroupreply (GroupName, Attribute, op, Value)
VALUES (“testgroup”,”Framed-Compression”,”==”,”Van-Jacobsen-TCP-IP”);

INSERT INTO radgroupreply (GroupName, Attribute, op, Value)
VALUES (“testgroup”,”Framed-Protocol”,”==”,”PPP”);
```

```
INSERT INTO radgroupreply (GroupName, Attribute, op, Value)
VALUES (“testgroup”,”Framed-MTU”,”==”,”1500”);

INSERT INTO radgroupreply (GroupName, Attribute, op, Value)
VALUES (“testgroup”,”Service-Type”,”==”,”Framed-User”);
```

Configuration Files

We finally need to change a few configuration files:

```
cd /etc/raddb
```

radiusd.conf

Find the “authorize” section (near the bottom) and add “sql” (no quotes) between “suffix” and “files”. Comment out “files” by putting a # in front of it. Do the same to “suffix”.

Find the “accounting” section (a bit underneath the authorize section) and place the word “sql” (without quotes” between “unix” and radutmp”

sql.conf

Find And modify the following fields:

```
#connect info
server = "localhost"
login = "root"
password = "rootpass" ß {change this to the root password of the MySQL
DB}
.
.
.
# Print all SQL statements when in debug mode (-x)
sqltrace = no ß {change this to yes, not necessary but useful for
debugging}
```

Starting the Radius Server

Start the RADIUS server:

```
radiusd –x (omitting the –x will run it as a background purpose, when
ready to deploy permanently you should omit the –x)
```

Let’s test the RADIUS server using NTRadPing as before but use the username and password of the test user listed in the SQL database. You should receive an “Access-Accept” response and assuming you correctly entered the entries into the radgroupreply

table you should also see the following in the Attribute Dump portion of the response in NTRadPing:

```
Framed-Compression=VJ-TCP/IP-Header
Framed Protocol=PPP
Framed-MTU=1500
Service-Type=Framed
```

Congratulations you now have a fully functional RADIUS server authenticating against a MySQL database, and storing the accounting data in the MySQL database!!!

Getting the HR-WRAP to Work with the RADIUS Server

Log into the HR-WRAP and execute these simple commands:

For simplicities' sake later ensure you can ping the radius server from the HR-WRAP:

```
ping {ip address of your RADIUS server}
CTRL-C to break when you are satisfied you can/can not talk to the
server
```

If you can not ping your server you must fix that first before continuing

```
/radius
add service=hotspot address={ip address of your RADIUS server}
secret={secret key you defined in the clients file of the RADIUS
server}

/ip hotspot aaa set use-radius=yes
```

You should now, as a hotspot client, be able to request any page and be directed to the login page as normal, if you login as an entry in the SQL database (username: radiustest, password: testpassword) you shold be authenticated no problem.

There are many ways to use RADIUS with and HR-WRAP, the common applications are authorizing associations based on MAC address, and PPP(oE/TP) user authorization. Both of these are very simple to do, but can cause some frustration as they are not well documented in the manual. Keep in mind that you will need to add definitions for RADIUS servers in the RADIUS table. Each definition in the RADIUS table (click the RADIUS menu in HSConfig) is for a specific server, and you can have each specific server authenticate for different types of services, such as HotSpot, PPP[oE/TP], and Wireless.

To authorize associations on an AP interface, first set up a RADIUS server with "Wireless" enabled, then you simply need to set "radius-mac-authentication=yes" in the

security profile for the AP. You can do this through HSConfig by going to the Wireless->Security Profiles tab, double clicking your profile and ticking the "RADIUS MAC Authentication" box. The HR-WRAP will submit the MAC address as the username in the format 00:11:22:33:44:55 with a **blank password**. Since the HR-WRAP submits a blank password, you will need to keep this in mind when developing your security systems.

To authorize PPP[oE/TP] sessions, first set up a RADIUS server with "PPP" enabled, then on the PPP menu click "Secrets" and "AAA", then check radius (At the console, /ppp aaa use-radius=yes). There are many supported attributes that allow you to do many useful things with PPP/RADIUS, such as individual client WEP keys, and per-user queue limits. Read more about the available radius attributes on the Radius Manual Page

Accounting

To view the Accounting database table for a user:

```
cd /usr/bin (if you aren't already there)
mysql –uroot –p{root password}
USE radius;
SELECT * FROM radacct;
```

- Note each login/logout pair is recorded as one row in the table

HauteRouterOS SNMP Service

Document revision: 1.7 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

SNMP is an application layer protocol. It is called simple because it works that way - the management station makes a request, and the managed device (SNMP agent) replies to this request. In SNMPv1 there are three main actions - Get, Set, and Trap. RouterOS supports only Get, which means that you can use this implementation only for network monitoring.

Hosts receive SNMP generated messages on UDP port 161 (except the trap messages, which are received on UDP port 162).

The HauteRouterOS supports:

- SNMPv1 only
- Read-only access is provided to the NMS (network management system)
- User defined communities are supported
- Get and GetNext actions
- No Set support
- No Trap support

Specifications

Packages required: ***system***, ***ppp****(optional)*
License required: *Level1*
Submenu level: ***/snmp***
Standards and Technologies: SNMP (RFC 1157)
Hardware usage: *Not significant*

Additional Resources

- http://www.ietf.org/rfc/rfc1157.txt
- http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/snmp.htm
- http://www.david-guerrero.com/papers/snmp/

SNMP Setup

Submenu level: ***/snmp***

Description

This section shows you how to enable the SNMP agent on HauteRouterOS.

Property Description

enabled (yes | no) - whether the SNMP service is enabled
contact (*text*; default: **""**) - contact information for the NMS
location (*text*; default: **""**) - location information for the NMS

Example

To enable the service, specifying some info:

```
[admin@Hautespot] snmp> set contact="admin@riga-2" location="3rd floor"
enabled="yes"
[admin@Hautespot] snmp> print
     enabled: yes
     contact: admin@riga-2
```

```
    location: 3rd floor
[admin@Hautespot] snmp>
```

SNMP Communities

Submenu level: ***/snmp community***

Description

The community name is a value in SNMPv1 header. It is like a 'username' for connecting to the SNMP agent. The default community for SNMP is **public**.

Property Description

name (*name*) - community name
address (*IP address mask*; default: **0.0.0.0/0**) - allow requests only from these addresses
0.0.0.0/0 - allow access for any address
read-access (yes | no; default: **yes**) - whether the read access is enabled for the community

Example

To view existing communities:

```
[admin@Hautespot] snmp community> print
 # NAME                                            ADDRESS            READ-
ACCESS
 0 public                                          0.0.0.0/0          yes
[admin@Hautespot] snmp community>
```

You can disable read access for the community **public**:

```
[admin@Hautespot] snmp community> set 0 read-access=no
[admin@Hautespot] snmp community> print
 # NAME                                            ADDRESS            READ-
ACCESS
 0 public                                          0.0.0.0/0          no
[admin@Hautespot] snmp community>
```

To add the community called **communa**, that is only accessible from the **159.148.116.0/24** network:

```
[admin@Hautespot] snmp community> add name=communa address=159.148.116.0/24
```

```
[admin@Hautespot] snmp community> print
# NAME                                            ADDRESS             READ-ACCESS
 0 public                                          0.0.0.0/0           no
 1 communa                                         159.148.116.0/24    no
[admin@Hautespot] snmp community>
```

Available OIDs

Description

OID stands for an object identifier, which is a data type specifying an authoritatively named object. An object identifier is a sequence of integers separated by decimal points. These integers traverse a tree structure, similar to the DNS or a Unix filesystem. There is an unnamed root at the top of the tree where the object identifiers start. All variables in the MIB start with the object identifier 1.3.6.1.2.1. Each node in the tree is also given a textual name. The names of the MIB variables are the numeric object identifiers, all of which begin with 1.3.6.1.2.1. You can use the SNMP protocol to get statistics from the router in these submenus:

- /interface
- /interface pc
- /interface wavelan
- /interface wireless
- /interface wireless registration-table
- /queue simple
- /queue tree
- /system identity
- /system license
- /system resource

Example

To see available OID values, just type **print oid**. For example, to see available OIDs in **/system resource**:

```
[admin@motors] system resource> print oid
             uptime: .1.3.6.1.2.1.1.3.0
    total-hdd-space: .1.3.6.1.2.1.25.2.3.1.5.1
     used-hdd-space: .1.3.6.1.2.1.25.2.3.1.6.1
       total-memory: .1.3.6.1.2.1.25.2.3.1.5.2
        used-memory: .1.3.6.1.2.1.25.2.3.1.6.2
           cpu-load: .1.3.6.1.2.1.25.3.3.1.2.1
[admin@motors] system resource>
```

Available MIBs

Description

The Management Information Base, or MIB, is the database of information maintained by the agent that the manager can query. You can download Hautespot MIB file

HauteRouterOS OID: enterprises.14988.1

RFC1493

dot1dBridge.dot1dBase.dot1dBaseBridgeAddress

dot1dBridge.dot1dStp.dot1dStpProtocolSpecification

dot1dBridge.dot1dStp.dot1dStpPriority

dot1dBridge.dot1dTp.dot1dTpFdbTable.dot1dTpFdbEntry.dot1dTpFdbAddress

dot1dBridge.dot1dTp.dot1dTpFdbTable.dot1dTpFdbEntry.dot1dTpFdbPort

dot1dBridge.dot1dTp.dot1dTpFdbTable.dot1dTpFdbEntry.dot1dTpFdbStatus

RFC2863

ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifName

ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifHCInOctets

ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifHCInUcastPkts

ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifHCOutOctets

ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifHCOutUcastPkts

RFC1213

interfaces.ifNumber

interfaces.ifTable.ifEntry.ifIndex

interfaces.ifTable.ifEntry.ifDescr

interfaces.ifTable.ifEntry.ifType

interfaces.ifTable.ifEntry.ifMtu

interfaces.ifTable.ifEntry.ifSpeed

interfaces.ifTable.ifEntry.ifPhysAddress

interfaces.ifTable.ifEntry.ifAdminStatus

interfaces.ifTable.ifEntry.ifOperStatus

interfaces.ifTable.ifEntry.ifLastChange

interfaces.ifTable.ifEntry.ifInOctets

interfaces.ifTable.ifEntry.ifInUcastPkts

interfaces.ifTable.ifEntry.ifInNUcastPkts

interfaces.ifTable.ifEntry.ifInDiscards

interfaces.ifTable.ifEntry.ifInErrors

interfaces.ifTable.ifEntry.ifInUnknownProtos

interfaces.ifTable.ifEntry.ifOutOctets

interfaces.ifTable.ifEntry.ifOutUcastPkts

interfaces.ifTable.ifEntry.ifOutNUcastPkts

interfaces.ifTable.ifEntry.ifOutDiscards

interfaces.ifTable.ifEntry.ifOutErrors

interfaces.ifTable.ifEntry.ifOutQLen

RFC2011

ip.ipForwarding

ip.ipDefaultTTL

ip.ipAddrTable.ipAddrEntry.ipAdEntAddr

ip.ipAddrTable.ipAddrEntry.ipAdEntIfIndex

ip.ipAddrTable.ipAddrEntry.ipAdEntNetMask

ip.ipAddrTable.ipAddrEntry.ipAdEntBcastAddr

ip.ipAddrTable.ipAddrEntry.ipAdEntReasmMaxSize

ip.ipNetToMediaTable.ipNetToMediaEntry.ipNetToMediaIfIndex

ip.ipNetToMediaTable.ipNetToMediaEntry.ipNetToMediaPhysAddress

ip.ipNetToMediaTable.ipNetToMediaEntry.ipNetToMediaNetAddress

ip.ipNetToMediaTable.ipNetToMediaEntry.ipNetToMediaType

RFC2096

ip.ipForward.ipCidrRouteNumber

ip.ipForward.ipCidrRouteTable.ipCidrRouteEntry.ipCidrRouteDest

ip.ipForward.ipCidrRouteTable.ipCidrRouteEntry.ipCidrRouteMask

ip.ipForward.ipCidrRouteTable.ipCidrRouteEntry.ipCidrRouteTos

ip.ipForward.ipCidrRouteTable.ipCidrRouteEntry.ipCidrRouteNextHop

ip.ipForward.ipCidrRouteTable.ipCidrRouteEntry.ipCidrRouteIfIndex

ip.ipForward.ipCidrRouteTable.ipCidrRouteEntry.ipCidrRouteType

ip.ipForward.ipCidrRouteTable.ipCidrRouteEntry.ipCidrRouteProto

ip.ipForward.ipCidrRouteTable.ipCidrRouteEntry.ipCidrRouteAge

ip.ipForward.ipCidrRouteTable.ipCidrRouteEntry.ipCidrRouteInfo

ip.ipForward.ipCidrRouteTable.ipCidrRouteEntry.ipCidrRouteNextHopAS

ip.ipForward.ipCidrRouteTable.ipCidrRouteEntry.ipCidrRouteMetric1

ip.ipForward.ipCidrRouteTable.ipCidrRouteEntry.ipCidrRouteMetric2

ip.ipForward.ipCidrRouteTable.ipCidrRouteEntry.ipCidrRouteMetric3

ip.ipForward.ipCidrRouteTable.ipCidrRouteEntry.ipCidrRouteMetric4

ip.ipForward.ipCidrRouteTable.ipCidrRouteEntry.ipCidrRouteMetric5

ip.ipForward.ipCidrRouteTable.ipCidrRouteEntry.ipCidrRouteStatus

Note that obsolete ip.ipRouteTable is also supported

RFC1213

system.sysDescr

system.sysObjectID

system.sysUpTime

system.sysContact

system.sysName

system.sysLocation

system.sysServices

RFC2790

host.hrSystem.hrSystemUptime

host.hrSystem.hrSystemDate

host.hrStorage.hrMemorySize

host.hrStorage.hrStorageTable.hrStorageEntry.hrStorageIndex

host.hrStorage.hrStorageTable.hrStorageEntry.hrStorageType

host.hrStorage.hrStorageTable.hrStorageEntry.hrStorageDescr

host.hrStorage.hrStorageTable.hrStorageEntry.hrStorageAllocationUnits

host.hrStorage.hrStorageTable.hrStorageEntry.hrStorageSize

host.hrStorage.hrStorageTable.hrStorageEntry.hrStorageUsed

CISCO-AAA-SESSION-MIB

Note that this MIB is supported only when **ppp** package is installed. It reports both **ppp** and **hotspot** active users

enterprises.cisco.ciscoMgmt.ciscoAAASessionMIB.casnMIBObjects.casnActive.casnActiveTableEntries

enterprises.cisco.ciscoMgmt.ciscoAAASessionMIB.casnMIBObjects.casnActive.casnActiveTable.casnActiveEntry.casnSessionId

enterprises.cisco.ciscoMgmt.ciscoAAASessionMIB.casnMIBObjects.casnActive.casnActiveTable.casnActiveEntry.casnUserId

enterprises.cisco.ciscoMgmt.ciscoAAASessionMIB.casnMIBObjects.casnActive.casnActiveTable.casnActiveEntry.casnIpAddr

RFC2863

ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifInMulticastPkts

ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifInBroadcastPkts

ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifOutMulticastPkts

ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifOutBroadcastPkts

ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifHCInMulticastPkts

ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifHCInBroadcastPkts

ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifHCOutMulticastPkts

ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifHCOutBroadcastPkts

ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifHighSpeed

RFC2790

host.hrStorage.hrStorageTable.hrStorageEntry.hrStorageAllocationFailures

Tools for SNMP Data Collection and Analysis

Description

MRTG (Multi Router Traffic Grapher) is the most commonly used SNMP monitor. For further information, see this link: http://people.ee.ethz.ch/~oetiker/webtools/mrtg/.

An example of using MRTG with Hautespot SNMP

Here is a example configuration file for MRTG to monitor a network interface traffic on HauteRouterOS. This is only an example file.

```
##################################################################
# Multi Router Traffic Grapher -- Sample Configuration File
##################################################################
# This file is for use with mrtg-2.5.4c

# Global configuration
```

```
WorkDir: /var/www/mrtg
WriteExpires: Yes

RunAsDaemon: Yes
Interval: 6
Refresh: 300

######################################################################
# System: HR-WRAPDX
# Description: RouterOS v2.9
# Contact: support@Hautespot.com
# Location: Hautespot main office
######################################################################


### Interface 'RemOffice'

Target[HR-WRAPDX]:
1.3.6.1.2.1.2.2.1.10.8&1.3.6.1.2.1.2.2.1.16.8:public@1.1.1.3
#SetEnv[HR-WRAPDX]: MRTG_INT_IP="1.1.1.3" MRTG_INT_DESCR="ether1"
MaxBytes[HR-WRAPDX]: 1250000
Title[HR-WRAPDX]: Traffic Analysis for HR-WRAPDX(1)
PageTop[HR-WRAPDX]: <H1>Traffic Analysis for HR-WRAPDX(1)</H1>

  <TABLE>
    <TR>
      <TD>System:</TD> <TD>HR-WRAPDX</TD>
         </TR>
    <TR>
           <TD>Maintainer:</TD> <TD>HauteSpot Support</TD>
         </TR>
    <TR>
           <TD>Description:</TD><TD>An Embedded Board</TD>
         </TR>
       <TR>
           <TD>ifType:</TD>     <TD>ethernetCSMACD(6)</TD>
         </TR>
         <TR>
           <TD>ifName:</TD>     <TD>RemOffice</TD>
         </TR>
         <TR>
           <TD>Max Speed:</TD>  <TD>1250.0 kBytes/s</TD>
         </TR>
         <TR>
           <TD>IP:</TD>         <TD>10.10.2.1</TD>
    </TR>
  </TABLE>

### Queue 'queue1'

Target[HR-WRAPDX_queue]:
1.3.6.1.4.1.14988.1.1.2.1.1.8.1&1.3.6.1.4.1.14988.1.1.2.1.1.9.1:public@1.1.1.3
#SetEnv[HR-WRAPDX_queue]: MRTG_INT_IP="1.1.1.3" MRTG_INT_DESCR="ether1"
MaxBytes[HR-WRAPDX_queue]: 100000
Title[HR-WRAPDX_queue]: Traffic Analysis for HR-WRAPDX(1_1)
PageTop[HR-WRAPDX_queue]: <H1>Traffic Analysis for HR-WRAPDX(1_1)</H1>

  <TABLE>
    <TR>
      <TD>System:</TD> <TD>HR-WRAPDX</TD>
```

```
        </TR>
    <TR>
          <TD>Maintainer:</TD> <TD>MicroTik Support</TD>
        </TR>
    <TR>
          <TD>Description:</TD><TD>An Embedded Board</TD>
        </TR>
       <TR>
          <TD>ifType:</TD>     <TD>ethernetCSMACD(6)</TD>
        </TR>
        <TR>
          <TD>ifName:</TD>     <TD>RemOffice</TD>
        </TR>
              <TR>
          <TD>queueName:</TD>     <TD>queue1</TD>
        </TR>
        <TR>
          <TD>Max Speed:</TD>  <TD>64.0 kBytes/s</TD>
        </TR>
        <TR>
          <TD>IP:</TD>         <TD>10.10.2.1</TD>
    </TR>
  </TABLE>
```

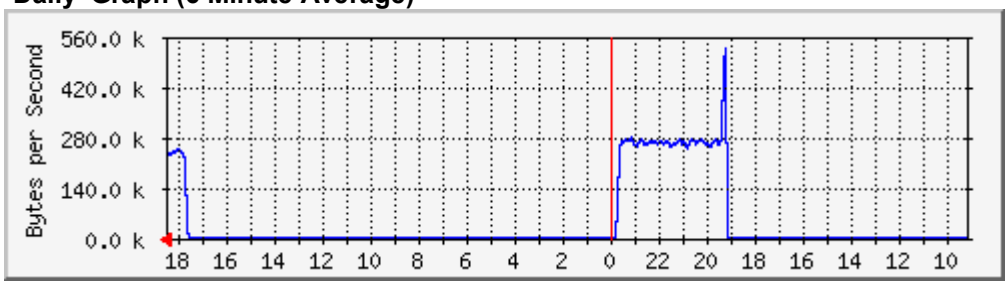
The output page of MRTG (interface part) should look like this:

Traffic Analysis for HR-WRAPDX(1)

System: HR-WRAPDX
Maintainer: HauteSpot Support
Description: An Embedded Board
ifType: ethernetCSMACD(6)
ifName: RemOffice
Max Speed: 1250.0 kBytes/s
IP: 10.10.2.1

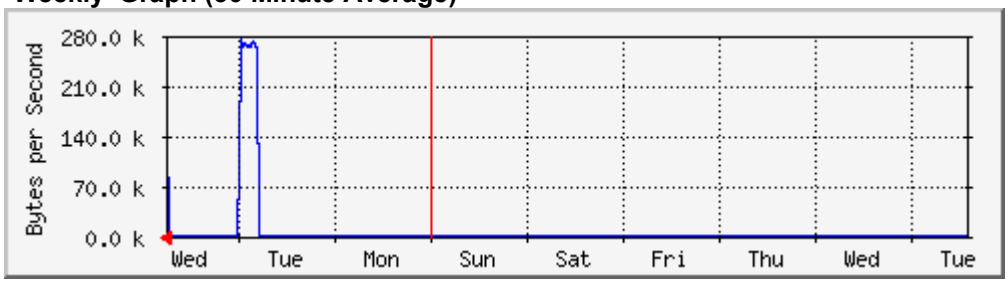
The statistics were last updated **Wednesday, 17 March 2004 at 18:33**,
at which time **'Hautespot'** had been up for **4:54:04**.

`Daily' Graph (6 Minute Average)



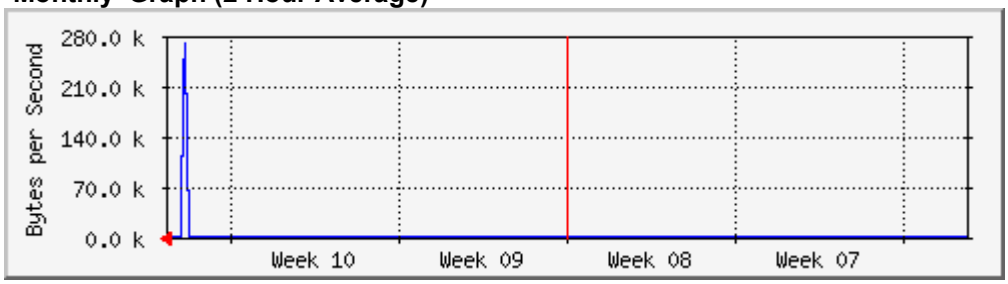
Max In:972.0 B/s (0.1%) Average In:125.0 B/s (0.0%) Current In:117.0 B/s (0.0%)
Max Out:526.9 kB/s (42.2%) Average Out:257.9 kB/s (20.6%) Current Out:236.5 kB/s (18.9%)

`Weekly' Graph (30 Minute Average)



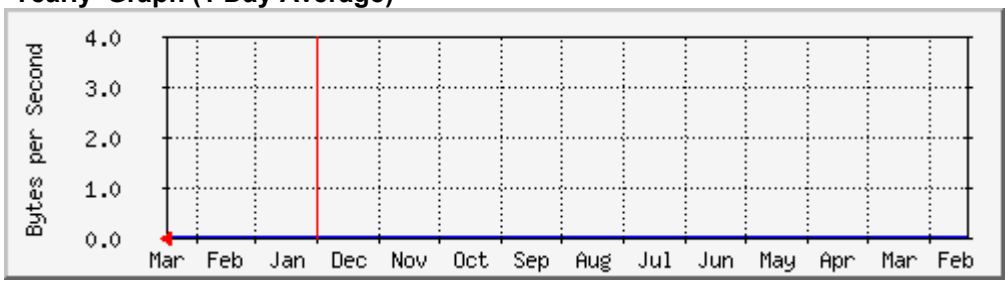
Max In:297.0 B/s (0.0%) Average In:108.0 B/s (0.0%) Current In:79.0 B/s (0.0%)
Max Out:274.8 kB/s (22.0%) Average Out:223.6 kB/s (17.9%) Current Out:163.4 kB/s (13.1%)

`Monthly' Graph (2 Hour Average)



Max In:112.0 B/s (0.0%) Average In:102.0 B/s (0.0%) Current In:0.0 B/s (0.0%)
Max Out:268.8 kB/s (21.5%) Average Out:210.0 kB/s (16.8%) Current Out:0.0 B/s (0.0%)

`Yearly' Graph (1 Day Average)



Max In:0.0 B/s (0.0%) Average In:0.0 B/s (0.0%) Current In:0.0 B/s (0.0%)
Max Out:0.0 B/s (0.0%) Average Out:0.0 B/s (0.0%) Current Out:0.0 B/s (0.0%)

GREEN ### Incoming Traffic in Bytes per Second

BLUE ### Outgoing Traffic in Bytes per Second

MRTG MULTI ROUTER TRAFFIC GRAPHER

2.9.29

Tobias Oetiker <oetiker@ee.ethz.ch>
and Dave Rand <dlr@bungi.com>

For more information read the MRTG documentation: Configuration Reference

HauteRouterOS SOCKS Proxy Server

Document revision: 1.3 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

This manual discusses the SOCKS proxy server which is implemented in RouterOS. HauteRouterOS supports SOCKS version 4.

Specifications

Packages required: ***system***
License required: *Level1*
Submenu level: ***/ip socks***
Standards and Technologies: SOCKS version 4
Hardware usage: *Not significant*

Description

SOCKS is a proxy server that allows TCP based application data to relay across the firewall, even if the firewall would block the packets. The SOCKS protocol is independent from application protocols, so it can be used for many services, e.g, WWW, FTP, TELNET, and others.

At first, an application client connects to the SOCKS proxy server, then the proxy server looks in its **access** list to see whether the client is permitted to access the remote application server or not, if it is permitted, the proxy server relies the packet to the application server and creates a connection between the application server and client.

Notes

Remember to configure your application client to use SOCKS version 4.

You should secure the SOCKS proxy using its access list and/or firewall to disallow access from outisde. Failing to secure the proxy server may introduce security issues to your network, and may provide a way for spammers to send junk mail through the router.

Additional Resources

- Information about SOCKS

SOCKS Configuration

Description

In this section you will learn how to enable the SOCKS proxy server and do its configuration.

Property Description

connection-idle-timeout (*time*; default: **2m**) - time after which idle connections are terminated
enabled (yes | no; default: **no**) - whether to enable or no the SOCKS proxy
max-connections (*integer*: 1..500; default: **200**) - maxumum number of simultaneous connections
port (*integer*: 1..65535; default: **1080**) - TCP port on which the SOCKS server listens for connections

Example

To enable SOCKS:

```
[admin@Hautespot] ip socks> set enabled=yes
[admin@Hautespot] ip socks> print
                    enabled: yes
                       port: 1080
    connection-idle-timeout: 2m
            max-connections: 200
```

```
[admin@Hautespot] ip socks>
```

Access List

Submenu level: ***/ip socks access***

Description

In the SOCKS access list you can add rules which will control access to SOCKS server. This list is similar to firewall lists.

Property Description

action (allow | deny; default: **allow**) - action to be performed for this rule
allow - allow packets, matching this rule to be forwarded for further processing
deny - deny access for packets, matching this rule
dst-address (*IP address*/*netmask*:*port*) - destination (server's) address
src-address (*IP address*/*netmask*:*port*) - source (client's) address for a packet

Active Connections

Submenu level: ***/ip socks connections***

Description

The Active Connection list shows all established TCP connections, which are maintained through the SOCKS proxy server.

Property Description

dst-address (*read-only: IP address*) - destination (application server) IP address
RX (*read-only: integer*) - bytes received
src-address (*read-only: IP address*) - source (application client) IP address
TX (*read-only: integer*) - bytes sent

Example

To see current TCP connections:

```
[admin@Hautespot] ip socks connections> print
 # SRC-ADDRESS               DST-ADDRESS               TX         RX
 0 192.168.0.2:3242          159.148.147.196:80        4847       2880
 1 192.168.0.2:3243          159.148.147.196:80        3408       2127
 2 192.168.0.2:3246          159.148.95.16:80          10172      25207
```

```
 3 192.168.0.2:3248          194.8.18.26:80             474        1629
 4 192.168.0.2:3249          159.148.95.16:80           6477       18695
 5 192.168.0.2:3250          159.148.95.16:80           4137       27568
 6 192.168.0.2:3251          159.148.95.16:80           1712       14296
 7 192.168.0.2:3258          80.91.34.241:80            314        208
 8 192.168.0.2:3259          80.91.34.241:80            934        524
 9 192.168.0.2:3260          80.91.34.241:80            930        524
10 192.168.0.2:3261          80.91.34.241:80            312        158
11 192.168.0.2:3262          80.91.34.241:80            312        158
[admin@Hautespot] ip socks connections>
```

Application Examples

FTP service through SOCKS server

Let us consider that we have a network **192.168.0.0/24** which is masqueraded, using a router with a public IP **10.1.0.104/24** and a private IP **192.168.0.1/24**. Somewhere in the network is an FTP server with IP address **10.5.8.8**. We want to allow access to this FTP server for a client in our local network with IP address **192.168.0.2/24**.

We have already masqueraded our local network:

```
[admin@Hautespot] ip firewall nat> print
Flags: X - disabled, I - invalid, D - dynamic
 0   chain=srcnat src-address=192.168.0.0/24 action=masquerade
[admin@Hautespot] ip firewall nat>
```

And the access to public FTP servers is denied in firewall:

```
[admin@Hautespot] ip firewall filter> print
Flags: X - disabled, I - invalid, D - dynamic
 0   chain=forward src-address=192.168.0.0/24 dst-address=:21 action=drop
[admin@Hautespot] ip firewall filter>
```

We need to enable the SOCKS server:

```
[admin@Hautespot] ip socks> set enabled=yes
[admin@Hautespot] ip socks> print
                    enabled: yes
                       port: 1080
    connection-idle-timeout: 2m
            max-connections: 200
[admin@Hautespot] ip socks>
```

Add access to a client with an IP address **192.168.0.2/32** to SOCKS access list, allow data transfer from FTP server to client (allow destionation ports from 1024 to 65535 for any IP address), and drop everything else:

```
[admin@Hautespot] ip socks access> add src-address=192.168.0.2/32 dst-
address=:21 action=allow
[admin@Hautespot] ip socks access> add dst-address=:1024-65535 action=allow
[admin@Hautespot] ip socks access> add action=deny
[admin@Hautespot] ip socks access> print
Flags: X - disabled
 0   src-address=192.168.0.2/32 dst-address=:21 action=allow
 1   dst-address=:1024-65535 action=allow
 2   action=deny
[admin@Hautespot] ip socks access>
```

That's all - the SOCKS server is configured. To see active connections and data transmitted and received:

```
[admin@Hautespot] ip socks connections> print
 # SRC-ADDRESS               DST-ADDRESS               TX         RX
 0 192.168.0.2:1238          10.5.8.8:21               1163       4625
 1 192.168.0.2:1258          10.5.8.8:3423             0          3231744
[admin@Hautespot] ip socks connections>
```

Note! In order to use SOCKS proxy server, you have to specify its IP address and port in your FTP client. In this case IP address would be **192.168.0.1** (router's/SOCKS server's local IP) and port **1080**.

Steps to long distance antenna alignment using the HR-WRAP

Long distance point to point or point to multipoint links require careful antenna alignment. An error of 1 or 2 degrees can mean the difference between a reliable link and a poor link. The HR-WRAP has several useful tools which can assist you in optimizing your antenna alignment.

1 Rough alignment – For long links, where you cannot physically see either end of the link, the best choice is to use a GPS receiver to determine the coordinates of each of the two ends of the link. Then use reciprocal compass headings to point the antennas at each other. Second choice is to use a map and compass to do the same.

In cases where you may be able to vaguely see the remote sites, use of a tool like a high power laser, strobe beacon, or even a large red flag, combined with binoculars may accomplish the task.

2. Mount your antennas with adjusting screws loose.

3. CHECK THE POLARITY OF YOUR ANTENNA. Panel antennas, parabolic antennas, and yagi antennas all have the ability to be mounted with either horizontal or vertical polarity. YOU MUST

MATCH THE POLARITY AT BOTH ENDS OF YOUR LINK. If your polarity is not correct at either end, the link will not work. This is a very common error.

4. Attach a HR-WRAP to each end of the link using a short pigtail connector. Make sure that the electrical connections to the pigtail are good at both ends (ie to the HR-WRAP and to the antenna). Follow the users manual to provide power to the unit and to connect to it using a lap top computer with a cross over cable. Open a session to the unit using HSConfig.

5. From within HSConfig, go to the Interface dialog and select the WLAN1 interface. Go to the Wireless dialog and set the band of the radio to the operating band that the link will function on. Obviously if you are using 2.4GHz antennas, use the 2.4GHz B Only mode. If you are using 5 GHz, use the 5Ghz mode.

6. Enable the WLAN1 Interface.

7. Repeat steps 4 to 6 for the other end of your link.

8. Now select the Align button to place the radio module into align mode. This will cause the radio to beacon constantly and to rapidly scan the channels of the band you selected for the radio. Make sure that Align mode is active on both ends of your link.

The following values are displayed in the alignment dialog:
address (*read-only: MAC address*) - MAC address of the remote host
avg-rxq (*read-only: integer*) - average signal strength of received packets since last display update on screen
correct (*read-only: percentage*) - how many undamaged packets were received
last-rx (*read-only: time*) - time in seconds before the last packet was received
last-tx (*read-only: time*) - time in seconds when the last TXQ info was received
rxq (*read-only: integer*) - signal strength of last received packet
ssid (*read-only: text*) - service set identifier
txq (*read-only: integer*) - the last received signal strength from our host to the remote one

9. Working with only one end of your link at a time, first move the antenna from right to left and stop when you see the maximum received signal from the remote end. Tighten down the horizontal alignment bolts so that the antenna cannot shift left or right. Then move the antenna slowly up and down looking for the best received signal indicator. Again, when the best signal is found, tighten down the bolts. The amount of movement to align the antenna may be very slight. Sometimes nothing more than a slight tap. Be patient and do not move too quickly or too much. A slight nudge is usually all you need.

10. Now repeat step 9 for the other end of the link. You may need to go back and fourth adjusting each end until you have the best tuned link. Do not, however, attempt to adjust both ends at the same time. Always leave one end static which you adjust the remote end.

As a general rule, you should be seeking a received signal strength of better than -82 for the link to be viable and ideally a link of -72 or better for best performance. These are just general guidelines and you should consider your link speed, rain fade and other criteria for your link.

11. Once your antennas are aligned, then you can either replace the HR-WRAP with whatever radio you plan on using, or bolt down your HR-WRAP for permanent service.

Troubleshoot areas

If your link quality is poor check the following:

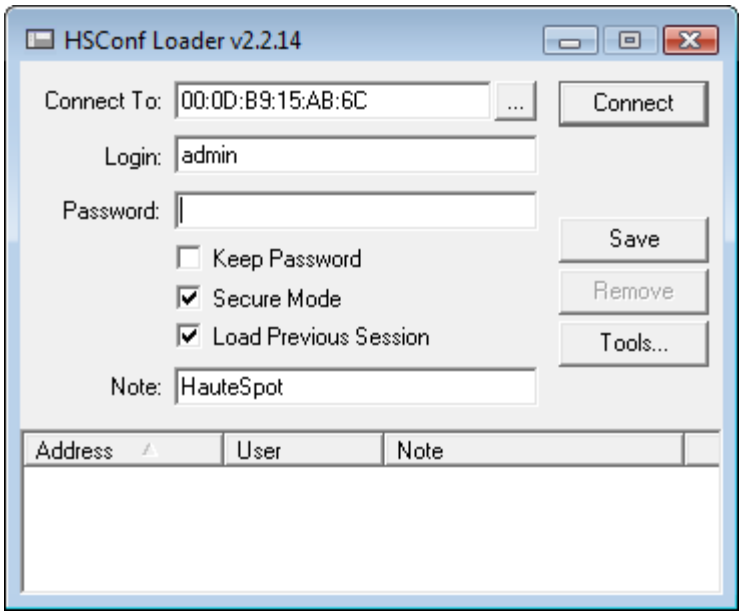
1. Are all your cable connectors tight.
2. Are your antennas still aligned
    a. Did wind blow your tower
    b. Were your bolts tight
3. Are there obstructions?
    a. You need to have both line of sight clearance and a clear Freznel zone. If you don't know what your Freznel zone clearance should be, consult our wireless installation calculator. You need to have Freznel zone clearance around your line of sight.
4. Is there interference?
    a. You can put the HR-WRAP into frequency utilization mode to sniff for traffic on your frequency and adjoining frequencies. This button is found in the Interface dialog for WLAN1. Remember that 802.11 channels overlap. If you have a strong source of interference on channel 3, it will affect channel 1, for example.
    b. Use a spectrum analyzer to look for non-802.11 sources of interference too.
    c. Change frequencies.
    d. Allow the HR-WRAP to run in RADAR Dynamic Frequency Selection mode and see what channel it selects. Then try using that channel instead of the one you are currently on.
5. Is there water in your connections?
    a. Use CRC 2-26 electrical lubricant to dry out your connections
    b. Wrap all connections with rubber splicing tape AND pvc electrical tape to protect from corrosion.
6. Do you have too much attenuation loss?
    a. Don't use a feeder cable longer than 20 feet without an amplifier
    b. Check gas vapor discharge devices for ground faults
    c. Check connectors for corrosion
7. Do you need an amp?
    a. Is your signal just too low?

Before you Start

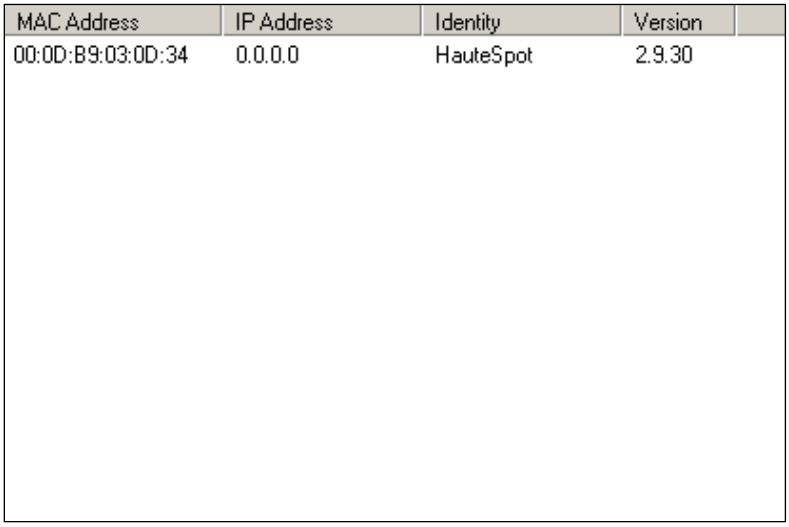
If you have forgotten all usernames and logins for the HR-WRAP unit you must return the unit back to HauteSpot Networks for a manual reset.

All outdoor HR-WRAP products will require the enclosure to be opened to access the serial connector. **This will void a unit's one year warranty**. Note that the HR-WRAP *indoor* models do not require the enclosure to be opened.

Most mis-configured/ inaccessible WRAP units can be recovered using HSconf's Mac Discovery option circled in the figure below. Please take all available steps of troubleshooting (eliminate power, ethernet etc. as potential issues) before moving forward.



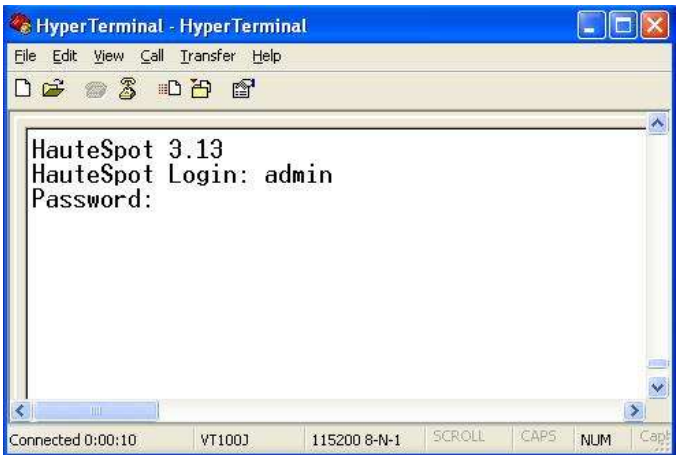
You can search for the router you wish to configure by clicking on the ... button which is located on the right of the Connect To: box. This will open a list of available routers found through layer 2 (MAC) detection. This is the most reliable way to connect to a router for configuration. Left click on the router you wish to manage, set the Login name to “admin” and leave the Password blank, and then click the connect button.



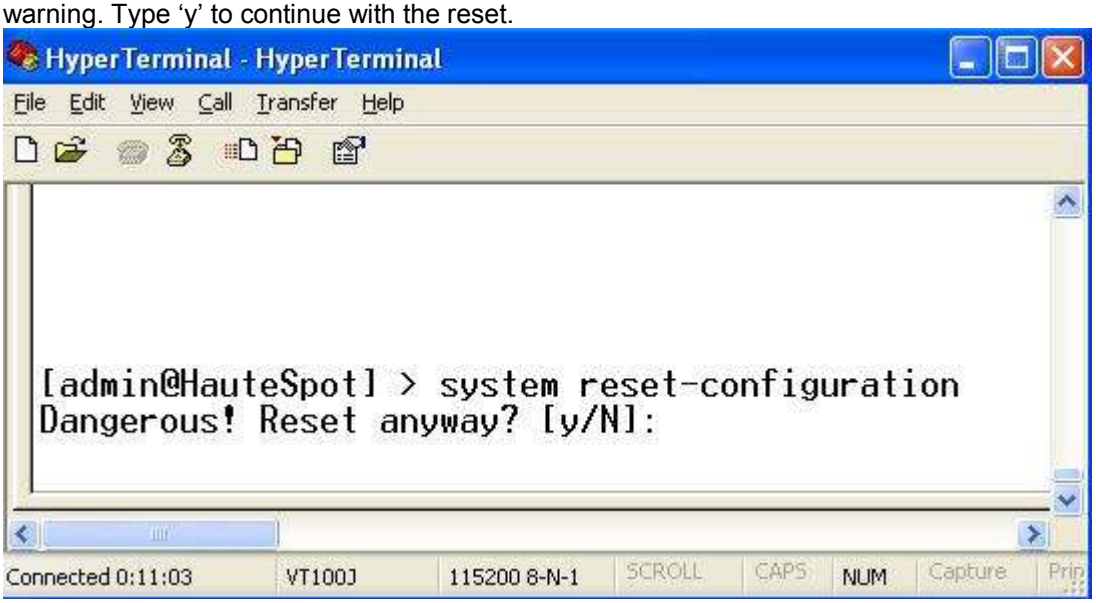
Steps to recover a completely corrupted system:

37 Attach the HR-WRAP to the computer using **a null modem cable** to the serial port.
38 Use a Terminal program such as HyperTerminal to configure. The setup for the serial connection should be 115200-8-N-1 with no flow control. Older (pre 2008) HR-WRAP products may use a 9600 baud.

3. Power on the system and wait one full minute before prompting (hitting any key). You may see text/symbols during this minute, this is normal. Once prompted after boot-up, the login will display. If the login is never displayed skip to step 6 to check for board status. There are two default usernames, admin and install. Both have no password at default.



4. Next type the command: system reset-configuration. You will be prompted with a certainty warning. Type 'y' to continue with the reset.

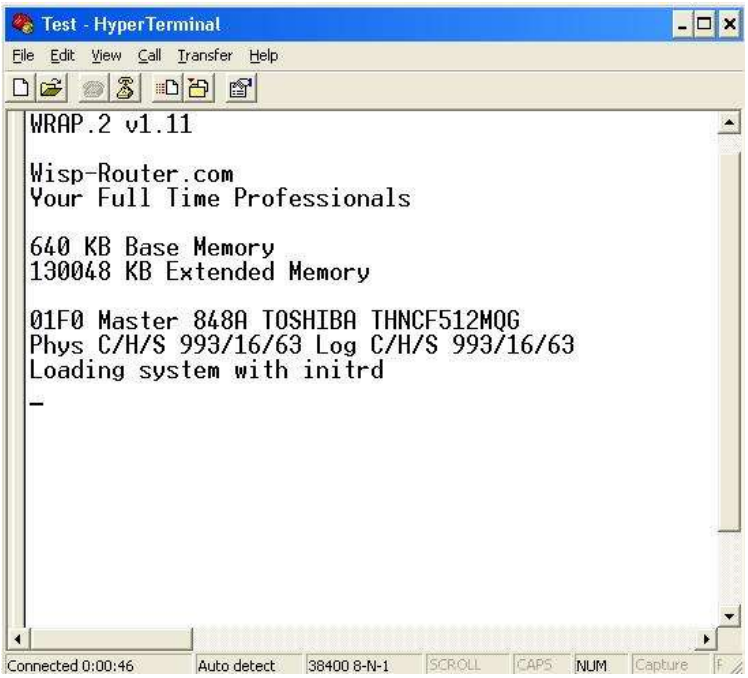


5. The unit will reboot using the default configuration. This includes an default IP of 192.168.1.1.

STEP 6 (diagnosing a bad board):

If you did receive a login after prompting, your HR-WRAP is suffering from one of two problems. The unit may have a damaged motherboard, which must be RMA'ed for repair. The unit also may be suffering from a corrupt flash disk. This can be repaired but will require the unit's enclosure to opened as well as the board removed from the enclosure.

To test the Motherboard status, simply change the baud rate of the terminal program to 38400 and reboot the unit. If the board is powering on the terminal will display:



If the board is verified to be working, the flash disk must be corrupt. Please download the newest flash disk image from the SOFTWARE section of our HR-WRAP support page. The steps to install the Disk Image are in the readme file once unzipped.

HauteRouterOS System Resource Management

Document revision: 2.2 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

HauteRouterOS offers several features for monitoring and managing the system resources.

Specifications

Packages required: ***system***
License required: *Level1*
Submenu level: ***/system***
Standards and Technologies: None
Hardware usage: *Not significant*

System Resource

Submenu level: ***/system resource***

Notes

In **monitor** command priotout the values for cpu usage and free memory are in percentage and kilobytes, respectively.

Example

To view the basic system resource status:

```
[admin@Hautespot] system resource> print
                      uptime: 04:32:41
                free-memory: 46488 kB
               total-memory: 62672 kB
                      model: HauteWRAP 230
                        cpu: Geode
                  cpu-load: 0
              free-hdd-space: 35873 kB
             total-hdd-space: 61972 kB
    write-sect-since-reboot: 2678
           write-sect-total: 408130
[admin@Hautespot] system resource>
```

To view the current system CPU usage and free memory:

```
[admin@Hautespot] > system resource monitor
       cpu-used: 0
    free-memory: 115676

[admin@Hautespot] >
```

IRQ Usage Monitor

Command name: ***/system resource irq print***

Description

IRQ usage shows which IRQ (Interrupt requests) are currently used by hardware.

Example

```
[admin@Hautespot] > system resource irq print
Flags: U - unused
   IRQ OWNER
   1   keyboard
```

```
   2   APIC
 U 3
   4   serial port
   5   [Ricoh Co Ltd RL5c476 II (#2)]
 U 6
 U 7
 U 8
 U 9
 U 10
   11  ether1
   12  [Ricoh Co Ltd RL5c476 II]
 U 13
   14  IDE 1
[admin@Hautespot] >
```

IO Port Usage Monitor

Command name: ***/system resource io print***

Description

IO usage shows which IO (Input/Output) ports are currently used by hardware.

Example

```
[admin@Hautespot] > system resource io print
 PORT-RANGE        OWNER
 0x20-0x3F         APIC
 0x40-0x5F         timer
 0x60-0x6F         keyboard
 0x80-0x8F         DMA
 0xA0-0xBF         APIC
 0xC0-0xDF         DMA
 0xF0-0xFF         FPU
 0x1F0-0x1F7       IDE 1
 0x2F8-0x2FF       serial port
 0x3C0-0x3DF       VGA
 0x3F6-0x3F6       IDE 1
 0x3F8-0x3FF       serial port
 0xCF8-0xCFF       [PCI conf1]
 0x4000-0x40FF     [PCI CardBus #03]
 0x4400-0x44FF     [PCI CardBus #03]
 0x4800-0x48FF     [PCI CardBus #04]
 0x4C00-0x4CFF     [PCI CardBus #04]
 0x5000-0x500F     [Intel Corp. 82801BA/BAM SMBus]
 0xC000-0xC0FF     [Realtek Semiconductor Co., Ltd. RTL-8139/8139C/8139C+]
 0xC000-0xC0FF     [8139too]
 0xC400-0xC407     [Cologne Chip Designs GmbH ISDN network controller [HFC-
PCI]
 0xC800-0xC87F     [Cyclades Corporation PC300/TE (1 port)]
 0xF000-0xF00F     [Intel Corp. 82801BA IDE U100]
```

```
[admin@Hautespot] >
```

USB Port Information

Command name: ***/system resource usb print***

Description

Shows all USB ports available for the router.

Property Description

device (*read-only: text*) - number of device
name (*read-only: text*) - name of the USB port
speed (*read-only: integer*) - bandwidth speed at which the port works
vendor (*read-only: text*) - vendor name of the USB device

Example

To list all available USB ports:

```
[admin@Hautespot] system resource usb> print
 # DEVICE VENDOR                  NAME                    SPEED
 0 1:1                            USB OHCI Root Hub       12 Mbps
[admin@Hautespot] system resource usb>
```

PCI Information

Command name: ***/system resource pci print***

Property Description

device (*read-only: text*) - number of device
irq (*read-only: integer*) - IRQ number which this device uses
name (*read-only: text*) - name of the USB port
vendor (*read-only: text*) - vendor name of the USB device

Example

To see PCI slot details:

```
[admin@Hautespot] system resource pci> print
```

```
 # DEVICE   VENDOR                       NAME                          IRQ
 0 00:13.0  Compaq                       ZFMicro Chipset USB (rev... 12
 1 00:12.5  National Semi                SC1100 XBus (rev: 0)
 2 00:12.4  National Semi                SC1100 Video (rev: 1)
 3 00:12.3  National Semi                SCx200 Audio (rev: 0)
 4 00:12.2  National Semi                SCx200 IDE (rev: 1)
 5 00:12.1  National Semi                SC1100 SMI (rev: 0)
 6 00:12.0  National Semi                SC1100 Bridge (rev: 0)
 7 00:0e.0  Atheros Communications       AR5212 (rev: 1)               10
 8 00:0d.1  Texas Instruments            PCI1250 PC card Cardbus ... 11
 9 00:0d.0  Texas Instruments            PCI1250 PC card Cardbus ... 11
10 00:0c.0  National Semi                DP83815 (MacPhyter) Ethe... 10
11 00:0b.0  National Semi                DP83815 (MacPhyter) Ethe... 9
12 00:00.0  Cyrix Corporation            PCI Master (rev: 0)
[admin@Hautespot] system resource pci>
```

Reboot

Command name: ***/system reboot***

Description

The system reboot is required when upgrading or installing new software packages. The packages are installed during the system shutdown.

The reboot process sends termination signal to all running processes, unmounts the file systems, and reboots the router.

Notes

Only users, which are members of groups with reboot privileges are permitted to reboot the router.

Reboot can be called from scripts, in which case it does not prompt for confirmation.

Example

```
[admin@Hautespot] > system reboot
Reboot, yes? [y/N]: y
system will reboot shortly
[admin@Hautespot] >
```

Shutdown

Command name: ***/system shutdown***

Description

Before turning the power off for the router, the system should be brought to halt. The shutdown process sends termination signal to all running processes, unmounts the file systems, and halts the router.

For most systems, it is necessary to wait approximately 30 seconds for a safe power down.

Notes

Only users, which are members of groups with reboot privileges are permitted to shutdown the router.

Shutdown can be called from scripts, in which case it does not prompt for confirmation.

Example

```
[admin@Hautespot] > system shutdown
Shutdown, yes? [y/N]: y
system will shutdown promptly
[admin@Hautespot] >
```

Router Identity

Submenu level: ***/system identity***

Description

The router identity is displayed before the command prompt. It is also used for DHCP client as 'host name' parameter when reporting it to the DHCP server.

Example

To view the router identity:

```
[admin@Hautespot] > system identity print
    name: "Hautespot"
[admin@Hautespot] >
```

To set the router identity:

```
[admin@Hautespot] > system identity set name=Gateway
[admin@Gateway] >
```

Date and Time

Submenu level: ***/system clock***

Property Description

date (*text*) - date in format "mm/DD/YYY"
dst-active (*read-only:* yes | no; default: **no**) - whether the Daylight Saving Time is currently acitve
time (*time*) - time in format "HH:MM:SS"
time-zone (*text*) - UTC timezone in format "+HH:MM" or "-HH:MM"

Notes

It is recommended that you reboot the router after time change to obviate the possible errors in time measurments and logging.

Date and time settings become permanent and effect BIOS settings.

If NTP update gives time shifted by 1 hour, although the time zone is set correctly, you may want to adjust the DST setting in **/system clock dst** menu.

Example

To view the current date and time settings:

```
[admin@Local] system clock> print
        time: 08:26:37
        date: nov/18/2004
   time-zone: +00:00
  dst-active: no
[admin@Local] system clock>
```

To set the system date and time:

```
[admin@Local] system clock> set date=nov/22/2022 time=11:10:21 time-zone=+0
[admin@Local] system clock> print
        time: 11:10:25
        date: nov/22/2022
   time-zone: +00:00
  dst-active: no
[admin@Local] system clock>
```

System Clock DST adjustment

Submenu level: ***/system clock dst***

Description

In most countries, a Daylight Saving Time regime is activated in spring and deactivated in autumn. This configuration menu provides DST adjustment facility, to drift the timezone according to your local legislation and practice.

Property Description

dst-delta (*text*; default: **+01:00**) - UTC timezone drift in format "+HH:MM" or "-HH:MM" to be added to the local timezone during DST period
dst-end (*date time*) - date and time when DST ends (when the delta is to be dropped).
dst-start (*date time*) - date and time when DST begins (when the delta is to be applied).

Example

To make DST zonechange active from **mar/27/2005 03:00:00** till **oct/30/2005 03:00:00**:

```
[admin@Hautespot] system clock dst> set dst-start="mar/27/2005 03:00:00" dst-
end="oct/30/2005 03:00:00"
[admin@Hautespot] system clock dst> print
  dst-delta: +01:00
  dst-start: mar/27/2005 03:00:00
    dst-end: oct/30/2005 03:00:00
[admin@Hautespot] system clock dst>
```

Configuration Change History

Specifications

Submenu level: ***/system history***, ***/undo***, ***/redo***

Description

The history of system configuration changes is held until the next router shutdown. The invoked commands can be 'undone' (in reverse order they have been invoked). The 'undone' commands may be 'redone' (in reverse order they have been 'undone').

Command Description

/redo - undoes previous '/undo' command
/system history print - print a list of last configuration changes, specifying whether the action can be undone or redone
/undo - undoes previous configuration changing command (except another '/undo' command)

Notes

Floating-undo actions are created within the current SAFE mode session. They are automatically converted to undoable and redoable when SAFE mode terminated successfully, and are all undone irreverively when SAFE mode terminated insuccessfully.

Undo command cannot undo commands past start of the SAFE mode.

Example

To show the list of configuration changes:

```
[admin@Hautespot] system history> print
Flags: U - undoable, R - redoable, F - floating-undo
   ACTION                                  BY                  POLICY
 U system time zone changed                admin               write
 U system time zone changed                admin               write
 U system time zone changed                admin               write
 U system identity changed                 admin               write
[admin@Hautespot] system clock>
```

What the **/undo** command does:

```
[admin@Hautespot] system history> print
Flags: U - undoable, R - redoable, F - floating-undo
   ACTION                                  BY                  POLICY
 R system time zone changed                admin               write
 U system time zone changed                admin               write
 U system time zone changed                admin               write
 U system identity changed                 admin               write
[admin@Hautespot] system clock>
```

System Note

Submenu level: ***/system note***

Description

System note feature allows you to assign arbitrary text notes or messages that will be displayed on each login right after banner. For example, you may distribute warnings between system administrators this way, or describe what does that particular router actually do. To configure system note, you may upload a plain text file named **sys-note.txt** on the router's FTP server, or, additionally, edit the settings in this menu

Property Description

note (*text*; default: **""**) - the note
show-at-login (yes | no; default: **yes**) - whether to show system note on each login

Notes

If you want to enter or edit multiline system note, you may need to use embedded text editor: `/system note edit note`

HauteRouterOS System Watchdog

Document revision: 1.2 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

System watchdog feature is needed to reboot the system in case of software failures.

Specifications

Packages required: ***system***
License required: *Level1*
Submenu level: ***/system watchdog***
Standards and Technologies:
Hardware usage: *Not significant*

Hardware Watchdog Management

Submenu level: ***/system watchdog***

Description

This menu allows to configure system to reboot on kernel panic, when an IP address does not respond, or in case the system has locked up. Software watchdog timer is used to provide the last option, so in very rare cases (caused by hardware malfunction) it can lock up by itself. There is a hardware watchdog device available in HauteRoute hardware, which can reboot the system in any case.

Property Description

reboot-on-failure (yes | no; default: **no**) - whether to reboot on kernel panic
watch-address (*IP address*; default: **none**) - if set, the system will reboot in case 6 sequental pings to the given IP address (sent once per 10 seconds) will fail
none - disable this option
watchdog-timer (yes | no; default: **no**) - whether to reboot if system is unresponsive for a minute
no-ping-delay (*time*; default: **5m**) - specifies how long after reboot not to test and ping **watch-address**. The default setting means that if **watch-address** is set and is not reachable, the router will reboot about every 6 minutes.
automatic-supout (yes | no; default: **yes**) - when software failure happens, a file named "autosupout.rif" is generated automatically. The previous "autosupout.rif" file is renamed to "autosupout.old.rif"
auto-send-supout (yes | no; default: **no**) - after the support output file is automatically generated, it can be sent by email
send-email-from (*text*; default: **""**) - e-mail address to send the support output file from. If not set, the value set in **/tool e-mail** is used
send-email-to (*text*; default: **""**) - e-mail address to send the support output file to
send-smtp-server (*text*; default: **""**) - SMTP server address to send the support output file through. If not set, the value set in **/tool e-mail** is used

Example

To make system generate a support output file and sent it automatically to **support@example.com** throught the **192.0.2.1**in case of a software crash:

```
[admin@Hautespot] system watchdog> set auto-send-supout=yes \
\... send-to-email=support@example.com send-smtp-server=192.0.2.1
[admin@Hautespot] system watchdog> print
  reboot-on-failure: yes
      watch-address: none
     watchdog-timer: yes
      no-ping-delay: 5m
   automatic-supout: yes
   auto-send-supout: yes
   send-smtp-server: 192.0.2.1
      send-email-to: support@example.com
[admin@Hautespot] system watchdog>
```

HauteRouterOS Terminal Console

Document revision: 1.0 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

The Terminal Console is used for accessing the HauteSpot Router's configuration and management features using text terminals, *id est* remote terminal clients or locally attached monitor and keyboard. The Terminal Console is also used for writing scripts. This manual describes the general console operation principles. Please consult the Scripting Manual on some advanced console commands and on how to write scripts.

Specifications

Packages required: ***system***
License required: *Level1*
Hardware usage: *Not significant*

Common Console Functions

Description

The console allows configuration of the router's settings using text commands. Although the command structure is similar to the Unix shell, you can get additional information about the command structure in the **Scripting Host and Complementary Tools** manual. Since there is a lot of available commands, they are split into groups organized in a way of hierarchical menu levels. The name of a menu level reflects the configuration information accessible in the relevant section, *exempli gratia* **/ip hotspot**.

In general, all menu levels hold the same commands. The difference is expressed mainly in command parameters.

Example

For example, you can issue the **/ip route print** command:

```
[admin@HauteSpot] > /ip route print
Flags: A - active, X - disabled, I - invalid, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, d - dynamic
 #     DST-ADDRESS        G GATEWAY         DISTANCE   INTERFACE
 0 ADC 1.1.1.0/24                                      isp2
 1 A S 2.2.2.0/24         r 1.1.1.2         0          isp2
 2 ADC 3.3.3.0/24                                      bonding1
 3 ADC 10.1.0.0/24                                     isp1
 4 A S 0.0.0.0/0          r 10.1.0.1        0          isp1

[admin@HauteSpot] >
```

Instead of typing **ip route** path before each command, the path can be typed only once to move into this particular branch of menu hierarchy. Thus, the example above could also be executed like this:

```
[admin@HauteSpot] > ip route
[admin@HauteSpot] ip route> print
Flags: A - active, X - disabled, I - invalid, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, d - dynamic
 #     DST-ADDRESS        G GATEWAY         DISTANCE   INTERFACE
 0 ADC 1.1.1.0/24                                      isp2
 1 A S 2.2.2.0/24         r 1.1.1.2         0          isp2
 2 ADC 3.3.3.0/24                                      bonding1
 3 ADC 10.1.0.0/24                                     isp1
 4 A S 0.0.0.0/0          r 10.1.0.1        0          isp1

[admin@HauteSpot] ip route>
```

Notice that the prompt changes in order to reflect where you are located in the menu hierarchy at the moment . To move to the top level again, type **/**:

```
[admin@HauteSpot] > /ip route
[admin@HauteSpot] ip route> /
[admin@HauteSpot] >
```

To move up one command level, type **..**:

```
[admin@HauteSpot] ip route> ..
[admin@HauteSpot] ip>
```

You can also use **/** and **..** to execute commands from other menu levels without changing the current level:

```
[admin@HauteSpot] ip route> /ping 10.0.0.1
10.0.0.1 ping timeout
2 packets transmitted, 0 packets received, 100% packet loss
[admin@HauteSpot] ip firewall nat> .. service-port print
```

```
Flags: X - disabled, I - invalid
 #   NAME                                                                 
PORTS
 0   ftp                                                                  21
 1   tftp                                                                 69
 2   irc                                                                  6667
 3 X h323
 4   quake3
 5   mms
 6   gre
 7   pptp
[admin@HauteSpot] ip firewall nat>
```

Lists and Item Names

Description

Lists

Many of the command levels operate with arrays of items: interfaces, routes, users etc. Such arrays are displayed in similarly looking lists. All items in the list have an item number followed by its parameter values.

To change parameters of an item, you have to specify it's number to the set command.

Item Names

Some lists have items that have specific names assigned to each. Examples are **interface** or **user** levels. There you can use item names instead of item numbers.

You do not have to use the **print** command before accessing items by name. As opposed to numbers, names are not assigned by the console internally, but are one of the items' properties. Thus, they would not change on their own. However, there are all kinds of obscure situations possible when several users are changing router's configuration at the same time. Generally, item names are more "stable" than the numbers, and also more informative, so you should prefer them to numbers when writing console scripts.

Notes

Item numbers are assigned by **print** command and are not constant - it is possible that two successive **print** commands will order items differently. But the results of last **print** commands are memorized and thus, once assigned, item numbers can be used even after **add**, **remove** and **move** operations (after **move** operation item numbers are moved with the items). Item numbers are assigned on per session basis, they will remain the same until you quit the console or until the next **print**

command is executed. Also, numbers are assigned separately for every item list, so **ip address print** would not change numbers for **interface** list.

Example

```
[admin@HauteSpot] interface> set 0 mtu=1200
ERROR: item number must be assigned by a print command
use print command before using an item number in a command
[admin@HauteSpot] interface> print
Flags: X - disabled, D - dynamic, R - running
 #    NAME                        TYPE            RX-RATE    TX-RATE    MTU
 0  R Public                      ether           0          0          1500
 1  R Local                       ether           0          0          1500
 2  R wlan1                       wlan            0          0          1500
[admin@HauteSpot] interface> set 0
disabled  mtu  name  rx-rate  tx-rate
[admin@HauteSpot] interface> set 0 mtu=1200
[admin@HauteSpot] interface> set wlan1 mtu=1300
[admin@HauteSpot] interface> print
Flags: X - disabled, D - dynamic, R - running
 #    NAME                        TYPE            RX-RATE    TX-RATE    MTU
 0  R Public                      ether           0          0          1200
 1  R Local                       ether           0          0          1500
 2  R wlan1                       wlan            0          0          1300
[admin@HauteSpot] interface>
```

Quick Typing

Description

There are two features in the console that help entering commands much quicker and easier - the [Tab] key completions, and abbreviations of command names. Completions work similarly to the **bash** shell in UNIX. If you press the [Tab] key after a part of a word, console tries to find the command within the current context that begins with this word. If there is only one match, it is automatically appended, followed by a space:

`/inte[Tab]_` becomes `/interface _`

If there is more than one match, but they all have a common beginning, which is longer than that what you have typed, then the word is completed to this common part, and no space is appended:

`/interface set e[Tab]_` becomes `/interface set ether_`

If you've typed just the common part, pressing the tab key once has no effect. However, pressing it for the second time shows all possible completions in compact form:

```
[admin@HauteSpot] > interface set e[Tab]_
[admin@HauteSpot] > interface set ether[Tab]_
[admin@HauteSpot] > interface set ether[Tab]_
ether1 ether5
[admin@HauteSpot] > interface set ether_
```

The [Tab] key can be used almost in any context where the console might have a clue about possible values - command names, argument names, arguments that have only several possible values (like names of items in some lists or name of protocol in firewall and NAT rules).You cannot complete numbers, IP addresses and similar values.

Another way to press fewer keys while typing is to abbreviate command and argument names. You can type only beginning of command name, and, if it is not ambiguous, console will accept it as a full name. So typing:

```
[admin@HauteSpot] > pi 10.1 c 3 si 100

equals to:

[admin@HauteSpot] > ping 10.0.0.1 count 3 size 100
```

Notes

Pressing [Tab] key while entering IP address will do a DNS lookup, instead of completion. If what is typed before cursor is a valid IP address, it will be resolved to a DNS name (reverse resolve), otherwise it will be resolved directly (i.e. to an IP address). To use this feature, DNS server must be configured and working. To avoid input lockups any such lookup will timeout after half a second, so you might have to press [Tab] several times, before the name is actually resolved.

It is possible to complete not only beginning, but also any distinctive substring of a name: if there is no exact match, console starts looking for words that have string being completed as first letters of a multiple word name, or that simply contain letters of this string in the same order. If single such word is found, it is completed at cursor position. For example:

```
[admin@HauteSpot] > interface x[TAB]_
[admin@HauteSpot] > interface export _
[admin@HauteSpot] > interface mt[TAB]_
[admin@HauteSpot] > interface monitor-traffic _
```

Additional Information

Description

Built-in Help

The console has a built-in help, which can be accessed by typing **?**. General rule is that help shows what you can type in position where the **?** was pressed (similarly to pressing [Tab] key twice, but in verbose form and with explanations).

Internal Item Numbers

You can specify multiple items as targets to some commands. Almost everywhere, where you can write the number of item, you can also write a list of numbers:

```
[admin@HauteSpot] > interface print
Flags: X - disabled, D - dynamic, R - running
  #    NAME                TYPE             MTU
  0  R ether1               ether            1500
  1  R ether2               ether            1500
  2  R ether3               ether            1500
  3  R ether4               ether            1500
[admin@HauteSpot] > interface set 0,1,2 mtu=1460
[admin@HauteSpot] > interface print
Flags: X - disabled, D - dynamic, R - running
  #    NAME                TYPE             MTU
  0  R ether1               ether            1460
  1  R ether2               ether            1460
  2  R ether3               ether            1460
  3  R ether4               ether            1500
[admin@HauteSpot] >
```

General Commands

Description

There are some commands that are common to nearly all menu levels, namely: **print, set, remove, add, find, get, export, enable, disable, comment, move**. These commands have similar behavior throughout different menu levels.

Command Description

print - shows all information that's accessible from particular command level. Thus, **/system clock print** shows system date and time, **/ip route print** shows all routes etc. If there's a list of items in current level and they are not read-only, i.e. you can change/remove them (example of read-only item list is **/system history**, which shows history of executed actions), then **print** command also assigns numbers that are used by all commands that operate with items in this list.

Input Parameters

from - applicable only to lists of items. The action is performed with all items in this list in the same order in which they are given.
brief - forces the **print** command to use tabular output form
detail - forces the **print** command to use **property=value** output form
count-only - shows the number of items

file - prints the contents of the specific submenu into a file. This file will be available in the router's ftp
interval - shows the output from the **print** command for every **interval** seconds
oid - prints the oid value, which is useful for SNMP
without-paging - prints the output without paging, to see printed output which does not fit in the screen, use [Shift]+[PgUp] key combination

It is possible to sort print output. Like this:

```
[admin@HauteSpot] interface> print type=ether
Flags: X - disabled, D - dynamic, R - running
 #    NAME                        TYPE             RX-RATE    TX-RATE    MTU
 0  R isp1                        ether            0          0          1500
 1  R isp2                        ether            0          0          1500
[admin@HauteSpot] interface>
```

set - allows you to change values of general parameters or item parameters. The **set** command has arguments with names corresponding to values you can change. Use **?** or double [Tab] to see list of all arguments. If there is a list of items in this command level, then **set** has one action argument that accepts the number of item (or list of numbers) you wish to set up. This command does not return anything.
add - this command usually has all the same arguments as **set**, except the action number argument. It adds a new item with values you have specified, usually to the end of list (in places where order is relevant). There are some values that you have to supply (like the interface for a new route), other values are set to defaults unless you explicitly specify them.
Input Parameters
copy-from - Copies an existing item. It takes default values of new item's properties from another item. If you do not want to make exact copy, you can specify new values for some properties. When copying items that have names, you will usually have to give a new name to a copy
place-before - places a new item before an existing item with specified position. Thus, you do not need to use the **move** command after adding an item to the list
disabled - controls disabled/enabled state of the newly added item(-s)
comment - holds the description of a newly created item
Return Values
unnamed - **add** command returns internal number of item it has added
remove - removes item(-s) from a list
Input Parameters
unnamed - contains number(-s) or name(-s) of item(-s) to remove.
move - changes the order of items in list where one is relevant. Item numbers after **move** command are left in a consistent, but hardly intuitive order, so it's better to resync them by using **print** after each **move** command.
Input Parameters
unnamed - first argument. Specifies the item(-s) being moved.
unnamed - second argument. Specifies the item before which to place all items being moved (they are placed at the end of the list if the second argument is omitted).
find - The find command has the same arguments as set, and an additional from argument which works like the from argument with the print command. Plus, find command has flag arguments like disabled, invalid that take values yes or no depending on the value of respective flag. To see all flags and their names, look at the top of print command's output. The find command returns internal numbers of all items that have the same values of arguments as specified.

edit - this command is in every place that has **set** command, it can be used to edit values of properties, *exempli gratia*:

```
[admin@HauteSpot] ip route> print
Flags: A - active, X - disabled, I - invalid, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, d - dynamic
 #     DST-ADDRESS        G GATEWAY         DISTANCE   INTERFACE
 0 ADC 1.1.1.0/24                                      isp2
 1 A S 2.2.2.0/24         r 1.1.1.2         0          isp2
 2 ADC 3.3.3.0/24                                      bonding1
 3 ADC 10.1.0.0/24                                     isp1
 4 A S 0.0.0.0/0          r 10.1.0.1        0          isp1
[admin@HauteSpot] ip route> edit 1 gateway
```

Safe Mode

Description

It is possible to change router configuration in a way that will make it not accessible except from local console. Usually this is done by accident, but there is no way to undo last change when connection to router is already cut. Safe mode can be used to minimize such risk.

Safe mode is entered by pressing **[Ctrl]+[X]**. To quit safe mode, press **[Ctrl]+[X]** again.

```
[admin@HauteSpot] ip route>[Ctrl]+[X]
[Safe Mode taken]

[admin@HauteSpot] ip route<SAFE>
```

Message **Safe Mode taken** is displayed and prompt changes to reflect that session is now in safe mode. All configuration changes that are made (also from other login sessions), while router is in safe mode, are automatically undone if safe mode session terminates abnormally. You can see all such changes that will be automatically undone tagged with an **F** flag in system history:

```
[admin@HauteSpot] ip route>
[Safe Mode taken]

[admin@HauteSpot] ip route<SAFE> add
[admin@HauteSpot] ip route<SAFE> /system history print
Flags: U - undoable, R - redoable, F - floating-undo
  ACTION                                   BY                POLICY
F route added                              admin             write
```

Now, if telnet connection is cut, then after a while (TCP timeout is **9** minutes) all changes that were made while in safe mode will be undone. Exiting session by **[Ctrl]+[D]**emphasis> also undoes all safe mode changes, while **/quit** does not.

If another user tries to enter safe mode, he's given following message:

```
[admin@HauteSpot] >
Hijacking Safe Mode from someone - unroll/release/don't take it [u/r/d]:
```

- **[u]** - undoes all safe mode changes, and puts the current session in safe mode.
- **[r]** - keeps all current safe mode changes, and puts current session in a safe mode. Previous owner of safe mode is notified about this:

```
• [admin@HauteSpot] ip firewall rule input
```

```
[Safe mode released by another user]
```

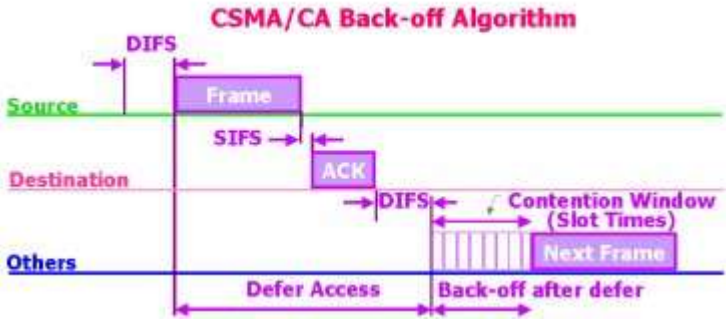
- **[d]** - leaves everything as-is.

If too many changes are made while in safe mode, and there's no room in history to hold them all (currently history keeps up to **100** most recent actions), then session is automatically put out of the safe mode, no changes are automatically undone. Thus, it is best to change configuration in small steps, while in safe mode. Pressing **[Ctrl]+[X]** twice is an easy way to empty safe mode action list.

Using HauteSpot TDMA-like protocol (TLP) for IP Video Over Wireless Multipoint to Point Links

IP video over wireless is a challenge. Because IP video needs a constant, reliable, high bit rate connection, without collisions and delay variability, wireless networks have traditionally been less than ideal. With the rise of IP encoding technologies like H.264, multi mega pixel cameras, license plate recognition, biometrics and the like, the need for real solutions for IP video over wireless is growing.

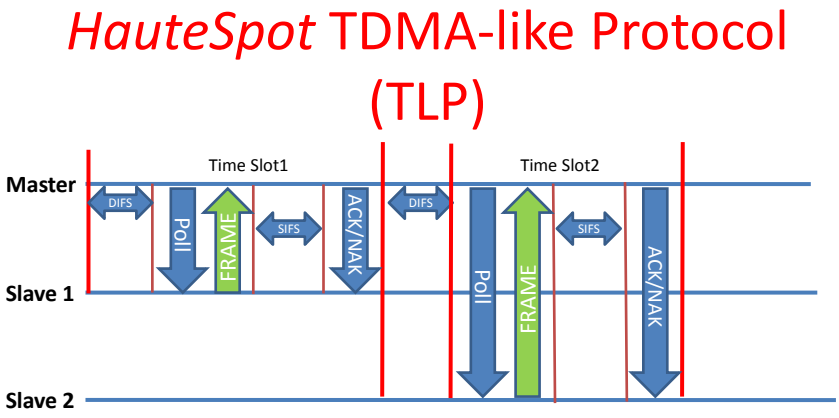
The 802.11 protocol set and derivatives, such as most mesh implementations, use a Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) protocol for wireless transmission. Basically every node on the wireless network attempts to send their data and if they don't receive an acknowledgement from the receiver, they assume that the data was lost and needs to be resent. The sender then backs off a random time period and tries to send again. This works fine for terminal data, text messaging, batch file transfers, etc. But when it comes to streaming data, statistical probability leads to retransmission storms. In other words, CSMA/CA is not very good for streaming media. Add to this the complexity and non-deterministic nature of mesh networks, and you quickly come to realize that CSMA/CA (ie 802.11 and mesh) is not the right solution for video.



Over the last year we have seen an ever increasing number of companies marketing products based on CSMA/CA as video over wireless. Customers see the price points or the claims of these vendors regarding ease of use and installation, and jump at these products only to be disappointed, particularly as more cameras need to be added.

Over two years ago HauteSpot Networks recognized the need for better wireless technologies for video and started development of our *HauteLINE* protocol for our *HauteROUTER HR-IXP* product family which replaced CSMA/CA with an point to point protocol which enabled the delivery of a RF link that could support up to 81Mbps of real throughput for video without jitter. The *HauteLINE* protocol was great for applications where you could cost justify point to point links for high throughput (like multi mega pixel video). We saw success with camera manufacturers who really needed our throughput, but we found that in surveillance there was another level of wireless support required for lower data rate multipoint to point operation with standard definition and higher compression H.264 cameras and encoders.

We have now released HauteSpot TDMA-like protocol, or “TLP”, a “Time Division Multiple Access” (TDMA) implementation for our *HauteWRAP HR-WRAP* product family that supports multipoint to point operation. This replaces CSMA/CA with a polling protocol that eliminates collisions and the randomness of back off and resends. Using this new protocol, multipoint to point networks can be created that scale much better for streaming media like video. We have tested the HSNstreme TDMA protocol in a variety of environments and found it to be an excellent performer.

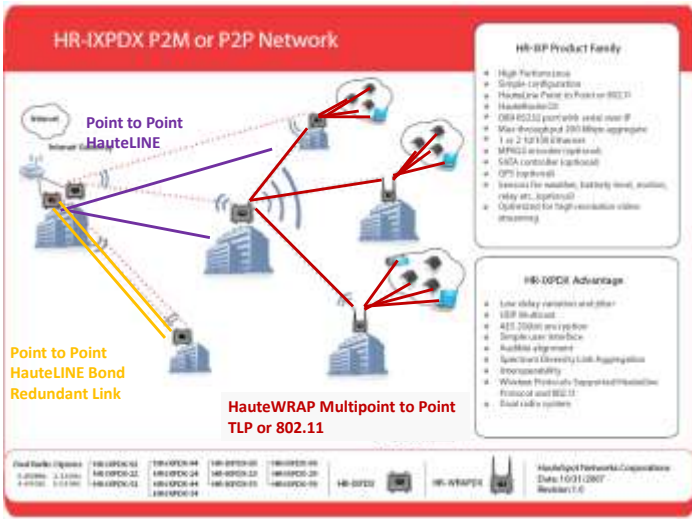


Maximum actual throughput data rates of the *HauteWRAP* with the TLP TDMA protocol are around 20Mbps, shared. This means that from a single base station (master) up to 7 remotes (slaves) radios can run at full resolution and full frame rate (assuming that each remote has an IP

camera requiring 2Mbps of nominal throughput), with no jitter. Most H.264 IP cameras fall into this data rate category. Reduce the frame rate, adjust the compression level, or change the resolution and you can get even more camera density.

In simple terms, we now offer a reliable wireless product which allows you to connect several remote cameras to one base station with excellent performance, no jitter like 802.11, and no delay like mesh.

WiMAX (802.16) uses the TDMA protocol too. However, because of the target markets for WiMAX, namely the telecom last mile service market, equipment is very expensive. A WiMAX base station for 200 subscribers can cost over $20K. WiMAX customer premise equipment can cost $500 to $1000 per subscriber. More important, WiMAX equipment is designed to be asymmetric favoring the forward or download direction (from the base station to the CPE), whereas video surveillance needs to be asymmetric favoring the return or upload direction. Most of the WiMAX customer premise equipment cannot support return channel rates that are as high as what our *HauteWRAP* product supports with the TLP TDMA protocol. And our products are much less expensive than WiMAX.



The *HauteWRAP* family can be used in a multipoint to point configuration to gather video streams to a central point and then the *HauteROUTER* family can be used to backhaul the video from the *HauteWRAP* to the primary monitoring location. This is the most economical and best performing wireless solution for IP video over wireless on the market today.

Of course our modular router design allows you to run your wireless network on a variety of frequencies from 700MHz up to 6GHz; our software can be customized for OEM applications; our product features can be extended to include environmental sensors; and our *HauteSHOT* line of video encoders, decoders and wireless bridge/decoders are the perfect complement to our wireless solutions by providing optimized encoding that maximizes your wireless network resources.

HauteSpot Networks has over 2500 *HauteWRAP* routers deployed already, with a field failure rate of less than 1.25%. Our installations include regional airports, military bases, rail yards, oil fields, natural gas pipelines, ship yards and ports, factories, municipalities, parking lots, college campus' and more. We work with our customers and system integrators to help develop workable, effective designs, and offer consulting services as well as excellent post sales technical support for our products.

About Hautespot Networks

From 20 yards to 20 miles, from CCTV to HD Broadcast production, HauteSpot Networks wireless technology allows any camera or weather station to be located virtually anywhere.

HauteSpot Networks' team of seasoned professionals from the embedded telecommunications and IP multimedia industries set out 2 years ago to develop, and is now currently selling, patent pending wireless technology specifically optimized for the needs of the rapidly growing video surveillance, environmental monitoring, and broadcast production markets.

Our existing customers such as the U.S. Military and Federal Law Enforcement already rely on our wireless solutions to carry high definition real-time video and weather data from remote locations back to wired broadband networks.

HauteSpot Networks' high performance modular wireless routing platform can be easily configured to meet the most demanding needs of HD broadcast, surveillance, environmental monitoring, and OEM customers. HauteSpot Networks' proprietary HauteLine™ protocol supports wireless video streaming at rates up to 65Mbps with low latency, no jitter, and predictable delay variation.

Founded in 2006 in San Luis Obispo California, HauteSpot Networks Corporation is privately held.

Contact:
Charlotte Chang
Director of Marketing Communications
HauteSpot Networks Corporation
P.O. Box 4016
San Luis Obispo, CA 93403
Phone: (805) 541-WISP (9477) Toll Free: (800) 541-5589 Fax: (805) 456-3829

HauteRouterOS Torch (Realtime Traffic Monitor)

Document revision: 1.8 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

Realtime traffic monitor may be used to monitor the traffic flow through an interface.

Specifications

Packages required: ***system***
License required: *Level1*
Submenu level: ***/tool***
Standards and Technologies: none
Hardware usage: *Not significant*

Description

Realtime Traffic Monitor called also torch is used for monitoring traffic that is going through an interface. You can monitor traffic classified by protocol name, source address, destination address, port. Torch shows the protocols you have chosen and mean transmitted and received data rate for each of them.

The Torch Command

Command name: ***/tool torch***

Property Description

interface (*name*) - the name of the interface to monitor
protocol (any | any-ip | ddp | egp | encap | ggp | gre | hmp | icmp | idpr-cmtp | igmp | ipencap | ipip | ipsec-ah | ipsec-esp | iso-tp4 | ospf | pup | rdp | rspf | st | tcp | udp | vmtp | xns-idp | xtp) - the name or number of the protocol
any - any ethernet or IP protocol
any-ip - any IP protocol
port (name | integer) - the name or number of the port
src-address (*IP address mask*) - source address and network mask to filter the traffic only with such an address, any source address: 0.0.0.0/0
dst-address (*IP address mask*) - destination address and network mask to filter the traffic only with such an address, any destination address: 0.0.0.0/0
average-seconds (*integer*: 1..10) - the average speed will be shown in the last average seconds
freeze-frame-interval (*time*) - time in seconds for which the screen output is paused

Notes

If there will be specific port given, then only **tcp** and **udp** protocols will be filtered, i.e., the name of the **protocol** can be **any**, **any-ip**, **tcp**, **udp**.

Except TX and RX, there will be only the field you've specified in command line in the command's output (e.g., you will get **PROTOCOL** column only in case if **protocol** property is explicitly specified).

Example

The following example monitors the traffic that goes through the **ether1** interface generated by **telnet** protocol:

```
[admin@Hautespot] tool> torch ether1 port=telnet
 SRC-PORT                    DST-PORT                    TX         RX
 1439                        23 (telnet)                 1.7kbps    368bps

[admin@Hautespot] tool>
```

To see what IP protocols are going through the **ether1** interface:

```
[admin@Hautespot] tool> torch ether1 protocol=any-ip
 PRO.. TX         RX
 tcp   1.06kbps   608bps
 udp   896bps     3.7kbps
 icmp  480bps     480bps
 ospf  0bps       192bps

[admin@Hautespot] tool>
```

To see what IP protocols are interacting with **10.0.0.144/32** host connected to the **ether1** interface:

```
[admin@Hautespot] tool> torch ether1 src-address=10.0.0.144/32 protocol=any
 PRO.. SRC-ADDRESS     TX         RX
 tcp   10.0.0.144      1.01kbps   608bps
 icmp  10.0.0.144      480bps     480bps

[admin@Hautespot] tool>
```

To see what tcp/udp protocols are going through the **ether1** interface:

```
[admin@Hautespot] tool> torch ether1 protocol=any-ip port=any
 PRO.. SRC-PORT                DST-PORT                TX         RX
 tcp   3430                    22 (ssh)                1.06kbps   608bps
 udp   2812                    1813 (radius-acct)      512bps     2.11kbps
 tcp   1059                    139 (netbios-ssn)       248bps     360bps
[admin@Hautespot] tool>
```

HauteRouterOS Traffic Flow

Document revision: 1.0 (Tuesday, October 30, 2007)

Applies to: HauteRouterOS V3.0

General Information

Specifications

Packages required: ***system***
License required: *Level1*
Submenu level: ***/ip traffic-flow***
Hardware usage: *Not significant*

Related Documents

- Cisco NetFlow
- NTop
- Integrating ntop with NetFlow

Description

Hautespot Traffic-Flow is a system that provides statistic information about packets which pass through the router. Besides network monitoring and accounting, system administrators can identify various problems that may occur in the network. With help of Traffic-Flow, it is possible to analyze and optimize the overall network performance. As Traffic-Flow is compatible with Cisco NetFlow, it can be used with various utilities which are designed for Cisco's NetFlow.

Traffic-Flow supports the following NetFlow formats:

- **version 1** - the first version of NetFlow data format, do not use it, unless you have to
- **version 5** - in addition to version 1, version 5 has the BGP AS and flow sequence number information included
- **version 9** - a new format which can be extended with new fields and record types thank's to its template-style design

General Configuration

Description

This section describes the basic configuration of Traffic-Flow.

Property Description

enabled (yes | no) - whether to enable traffic-flow service or not
interfaces (*name*) - names of those interfaces which will be used to gather statistics for traffic-flow. To specify more than one interface, separate them with a comma (",")

cache-entries (1k | 2k | 4k | 8k | 16k | 32k | 64k | 128k | 256k | 512k; default: **1k**) - number of flows which can be in router's memory simultaneously
active-flow-timeout (*time*; default: **30m**) - maximum life-time of a flow
inactive-flow-timeout (*time*; default: **15s**) - how long to keep the flow active, if it is idle

Traffic-Flow Target

Description

With Traffic-Flow targets we specify those hosts which will gather the Traffic-Flow information from router.

Property Description

address (*IP address*:*port*) - IP address and port (UDP) of the host which receives Traffic-Flow statistic packets from the router
v9-template-refresh (*integer*; default: **20**) - number of packets after which the template is sent to the receiving host (only for NetFlow version 9)
v9-template-timeout - after how long to send the template, if it has not been sent
version (1 | 5 | 9) - which version format of NetFlow to use

Application Examples

Traffic-Flow Example

This example shows how to configure Traffic-Flow on a router

1. Enable Traffic-Flow on the router:

```
[admin@Hautespot] ip traffic-flow> set enabled=yes
[admin@Hautespot] ip traffic-flow> print
                 enabled: yes
              interfaces: all
           cache-entries: 1k
     active-flow-timeout: 30m
   inactive-flow-timeout: 15s
```

```
[admin@Hautespot] ip traffic-flow>
```

9. Specify IP address and port of the host, which will receive Traffic-Flow packets:

```
[admin@Hautespot] ip traffic-flow target> add address=192.168.0.2:2055
\
\... version=9
[admin@Hautespot] ip traffic-flow target> print
Flags: X - disabled
```

```
14.  #    ADDRESS                VERSION
15.  0    192.168.0.2:2055       9
```

```
[admin@Hautespot] ip traffic-flow target>
```

Now the router starts to send packets with Traffic-Flow information.

Some screenshots from NTop program, which has gathered Traffic-Flow information from our router and displays it in nice graphs and statistics. For example, where what kind of traffic has flown:

Host Information

Traffic Unit: [**Bytes**] [Packets]

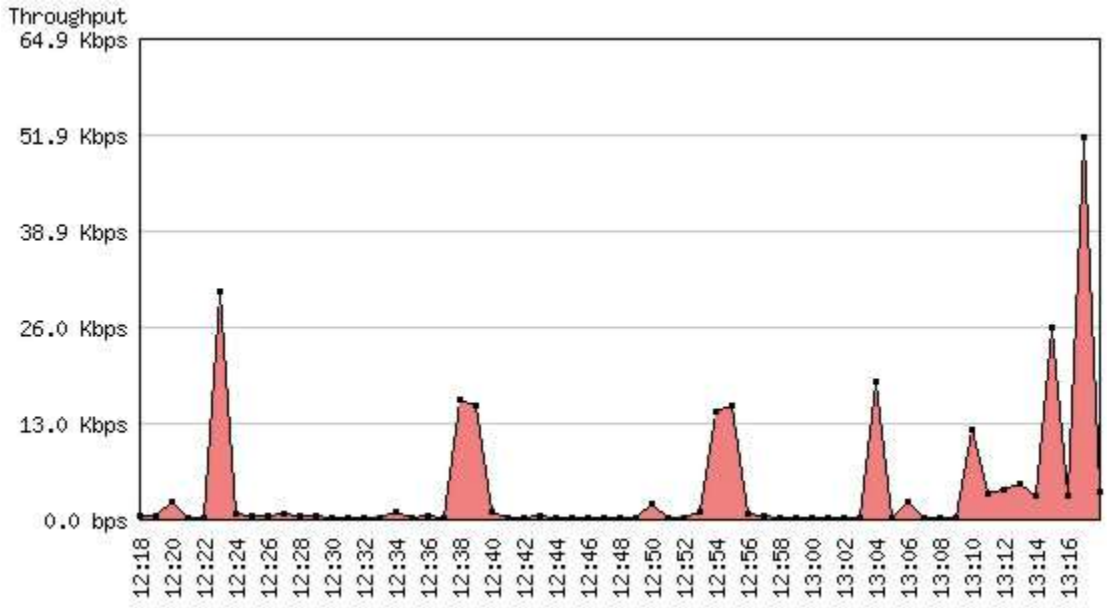
Host	Domain	IP Address	MAC Address	Other Name(s)	Bandwidth	Host Contacts	Age/Inactivity		AS
10.5.7.4		10.5.7.4				17	14 days 0:37:58	5 sec	
81.94.227.50		81.94.227.50				2	14 days 0:33:02	5:01	
255.255.255.255		255.255.255.255				6623	14 days 0:37:59	0 sec	
3.3.3.3		3.3.3.3				1	14 days 0:35:16	48 sec	
192.168.10.11		192.168.10.11				3	14 days 0:37:46	16 sec	
192.168.1.1		192.168.1.1				1	14 days 0:37:16	35 sec	
192.168.10.10		192.168.10.10				3	14 days 0:37:46	16 sec	
1120730533.383		10.5.5.3				1	14 days 0:36:29	39 sec	
webproxy.mt.lv		10.5.5.1				3	14 days 0:36:15	47 sec	
1120730600.335		10.5.5.2				1	14 days 0:35:16	48 sec	
dator1		10.5.5.111				2	14 days 0:35:02	33 sec	
daces		10.5.5.124				4	14 days 0:37:18	36 sec	
10.5.5.50		10.5.5.50				3	14 days 0:37:16	40 sec	

Top three hosts by upload and download each minute:

Network Load Statistics Matrix

Sampling Period	Average Thpt	Top Hosts Sent Thpt		Top Hosts Rcvd Thpt	
13:16 - 13:17	4.0 Kbps	10.5.7.4	872.0 bps	10.5.7.4	1.1 Kbps
		159.148.172.197	648.0 bps	195.13.237.141	640.0 bps
		10.5.7.1	640.0 bps	0.0.0.0	504.0 bps
13:15 - 13:16	51.9 Kbps	159.148.147.196	91.9 Kbps	10.5.7.14	91.9 Kbps
		10.5.7.14	3.4 Kbps	159.148.147.196	3.4 Kbps
		10.5.7.1	664.0 bps	10.5.7.4	1.1 Kbps
13:14 - 13:15	3.5 Kbps	10.5.7.4	856.0 bps	10.5.7.4	1.1 Kbps
		10.5.7.1	624.0 bps	195.13.237.141	624.0 bps
		195.13.237.141	608.0 bps	0.0.0.0	496.0 bps
13:13 - 13:14	26.2 Kbps	159.148.172.197	33.6 Kbps	10.5.54.1	33.6 Kbps
		10.5.54.1	968.0 bps	159.148.172.197	968.0 bps
		10.5.7.4	752.0 bps	192.168.10.10	48.0 bps
13:12 - 13:13	3.2 Kbps	10.5.7.4	1.8 Kbps	10.5.7.4	2.3 Kbps
		195.13.237.141	1.3 Kbps	195.13.237.141	1.3 Kbps
		192.168.10.10	960.0 bps	0.0.0.0	1.0 Kbps
13:11 - 13:12	4.9 Kbps	10.5.7.4	840.0 bps	10.5.7.4	1.1 Kbps
		195.13.237.141	624.0 bps	195.13.237.141	640.0 bps
		192.168.10.10	400.0 bps	0.0.0.0	480.0 bps

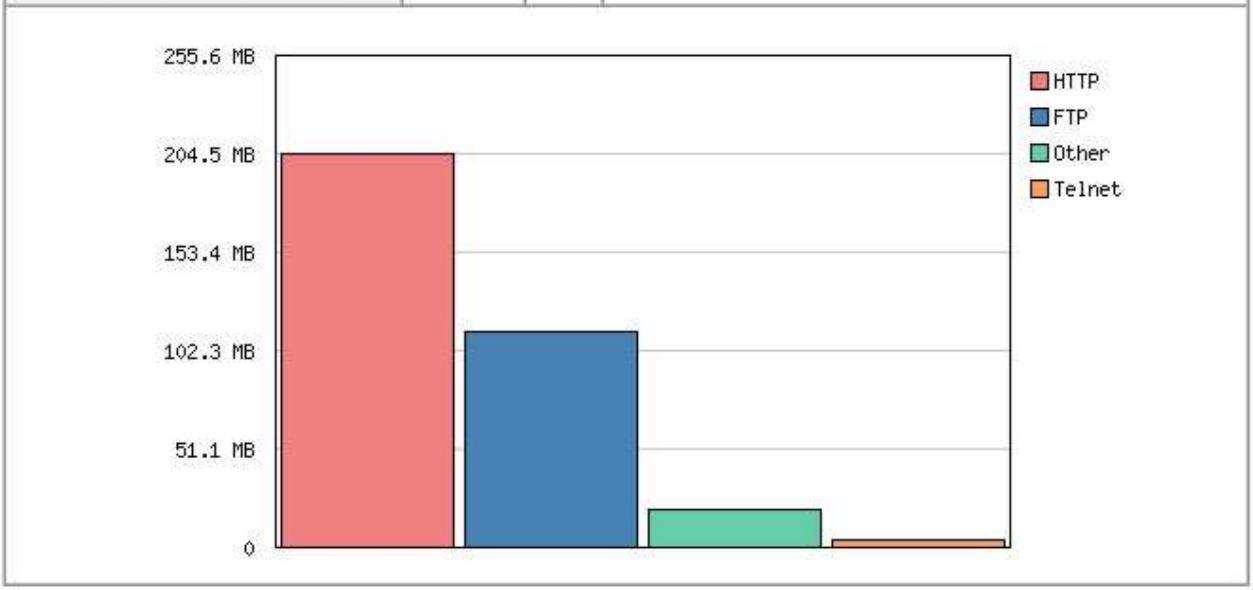
Overall network load each minute:



Traffic usage by each protocol:

Global TCP/UDP Protocol Distribution

TCP/UDP Protocol	Data	Percentage
FTP	112.3 MB	32%
HTTP	204.5 MB	59%
DNS	124.1 KB	0%
Telnet	4.5 MB	1%
NBios-IP	1.0 MB	0%
Mail	1.7 MB	0%
DHCP-BOOTP	22.0 KB	0%
Messenger	0.3 KB	0%
Other TCP/UDP-based Protocols	17.0 MB	4%



Transparently Bridge two Networks using MPLS

Overview

This a very basic example how to enable MPLS, establish VPLS tunnel between two wireless links and use it to transparently bridge two networks.

There are several other ways to create this type of setup:

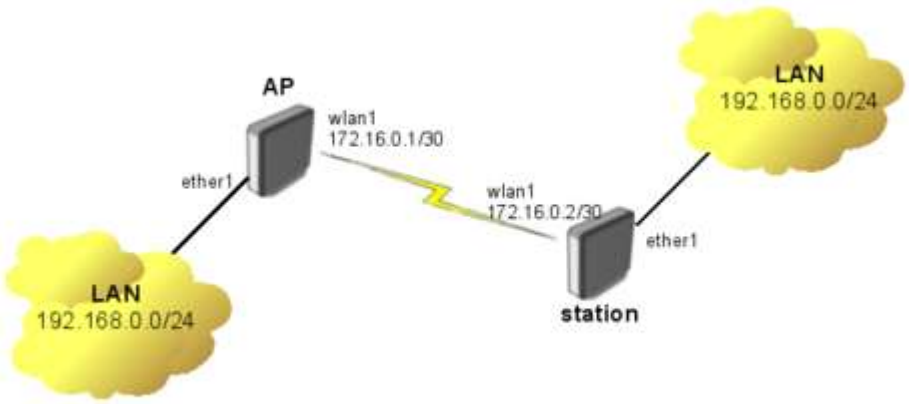
- Bridge using WDS
- Bridge using EoIP

The MPLS/VPLS approach has some advantages:

- VPLS tunnel is about 60% faster and less overhead than EoIP tunnel
- 802.11n speed is limited over WDS bridges, this method doesn't have such limitations

Configuration

Let us assume the following network setup:



Note: For this setup to work in HauteRouterOS 3.x, **routing** and **mpls-test** package must be installed. These features are included by default in 4.x.

AP

```
# --configure wireless access point--
/interface wireless
set wlan1 disabled=no ssid=MPLS frequency=5180 band=5ghz mode=bridge

# --configure IP--
/ip address
add address=172.16.0.1/30 interface=wlan1

# --enable LDP--
/mpls ldp
set enabled=yes lsr-id=172.16.0.1 transport-address=172.16.0.1
/mpls ldp interface
add interface=wlan1
```

```
# --configure VPLS tunnel--
/interface vpls
add name=vpls1 remote-peer=172.16.0.2 vpls-id=1:1 disabled=no

# --add bridge and bridge ports --
/interface bridge add name=bridge1
/interface bridge port
add bridge=bridge1 interface=ether1
add bridge=bridge1 interface=vpls1
```

station

```
# --configure wireless access point--
/interface wireless
set wlan1 disabled=no ssid=MPLS band=5ghz mode=station

# --configure IP--
/ip address
add address=172.16.0.2/30 interface=wlan1

# --enable LDP--
/mpls ldp
set enabled=yes lsr-id=172.16.0.2 transport-address=172.16.0.2
/mpls ldp interface
add interface=wlan1

# --configure VPLS tunnel--
/interface vpls
add name=vpls1 remote-peer=172.16.0.1 vpls-id=1:1 disabled=no

# --add bridge and bridge ports --
/interface bridge add name=bridge1
/interface bridge port
add bridge=bridge1 interface=ether1
add bridge=bridge1 interface=vpls1
```

Confirm that LDP is running

```
[admin@HauteSpot] /mpls ldp neighbor> print
Flags: X - disabled, D - dynamic, O - operational, T - sending-
targeted-hello, V - vpls
 #      TRANSPORT    LOCAL-TRANSPORT PEER           SEND-TARGETED
ADDRESSES
 0 DOTV 172.16.0.2   172.16.0.1      172.16.0.2:0   no
172.16.0.2

[admin@HauteSpot] /mpls ldp neighbor> .. .. forwarding-table print
Flags: L - ldp, V - vpls, T - traffic-eng
 #   IN-LABEL        OUT-LABELS      DESTINATION    INTERFACE
NEXTHOP
 0   expl-null
 1 V 18                              vpls1
```

Confirm that VPLS tunnel is established:

```
[admin@HauteSpot] /interface vpls> monitor vpls1 once
       remote-label: 17
        local-label: 18
      remote-status:
  transport-nexthop: 172.16.0.2
     imposed-labels: 17
```

Connect more stations to the same Bridge

It is possible to connect more than one Station to the same Bridge, simply creating more VPLS tunnels (one for Station):

AP

```
# --change wireless access point mode to AP Bridge--
/interface wireless
set wlan1 mode=ap-bridge

# --configure second VPLS tunnel--
/interface vpls
add name=vpls2 remote-peer=172.16.0.3 vpls-id=2:2 disabled=no

# --add vpls2 to bridge --

/interface bridge port
add bridge=bridge1 interface=vpls2
```

2nd Station

```
# --configure wireless access point--
/interface wireless
set wlan1 disabled=no ssid=MPLS band=5ghz mode=station

# --configure IP--
/ip address
add address=172.16.0.3/30 interface=wlan1

# --enable LDP--
/mpls ldp
set enabled=yes lsr-id=172.16.0.3 transport-address=172.16.0.3
/mpls ldp interface
add interface=wlan1

# --configure VPLS tunnel--
/interface vpls
add name=vpls1 remote-peer=172.16.0.1 vpls-id=2:2 disabled=no

# --add bridge and bridge ports --
```

```
/interface bridge add name=bridge1
/interface bridge port
add bridge=bridge1 interface=ether1
add bridge=bridge1 interface=vpls1
```

Note

When router encapsulates Ethernet frame to forward over VPLS pseudowire, it checks if packet size + VPLS CW + MPLS labels exceeds MPLS MTU of outgoing interface. If it does, VPLS will fragment frame. In this example 1514byte layer2 packets are forwarded over VPLS, router adds CW (8bytes) and one MPLS tag (4bytes) it means that to avoid fragmentation MPLS MTU must be increased to 1526

```
/mpls interface set 0 mpls-mtu=1526
```

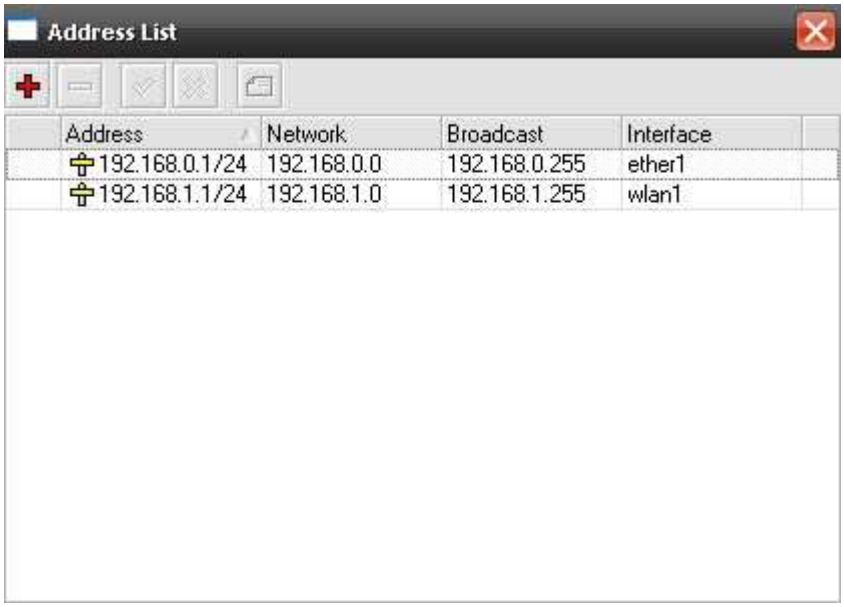
Ethernet port should be capable of sending 1526 byte L2MTU packets.

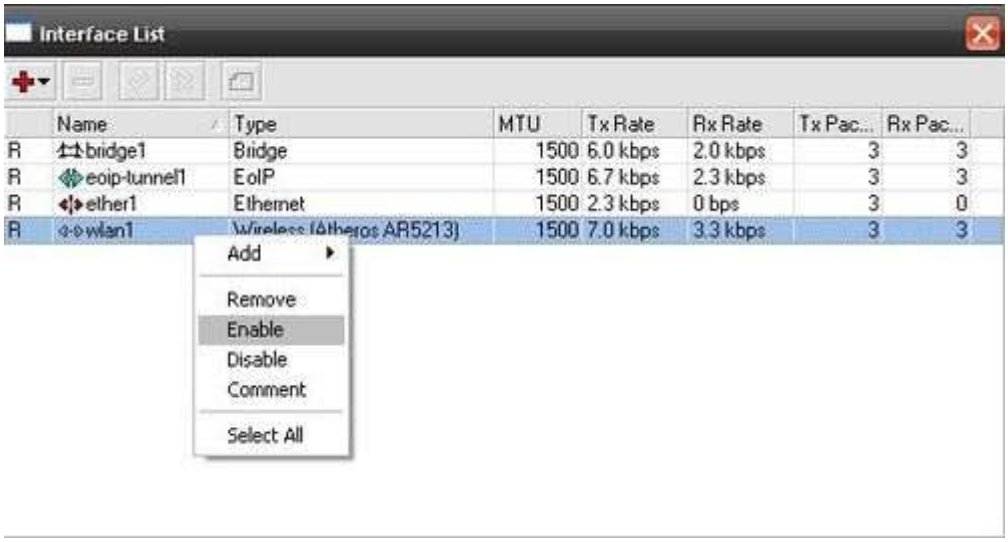
Transparently Bridge two Networks without using WDS (EoIP)

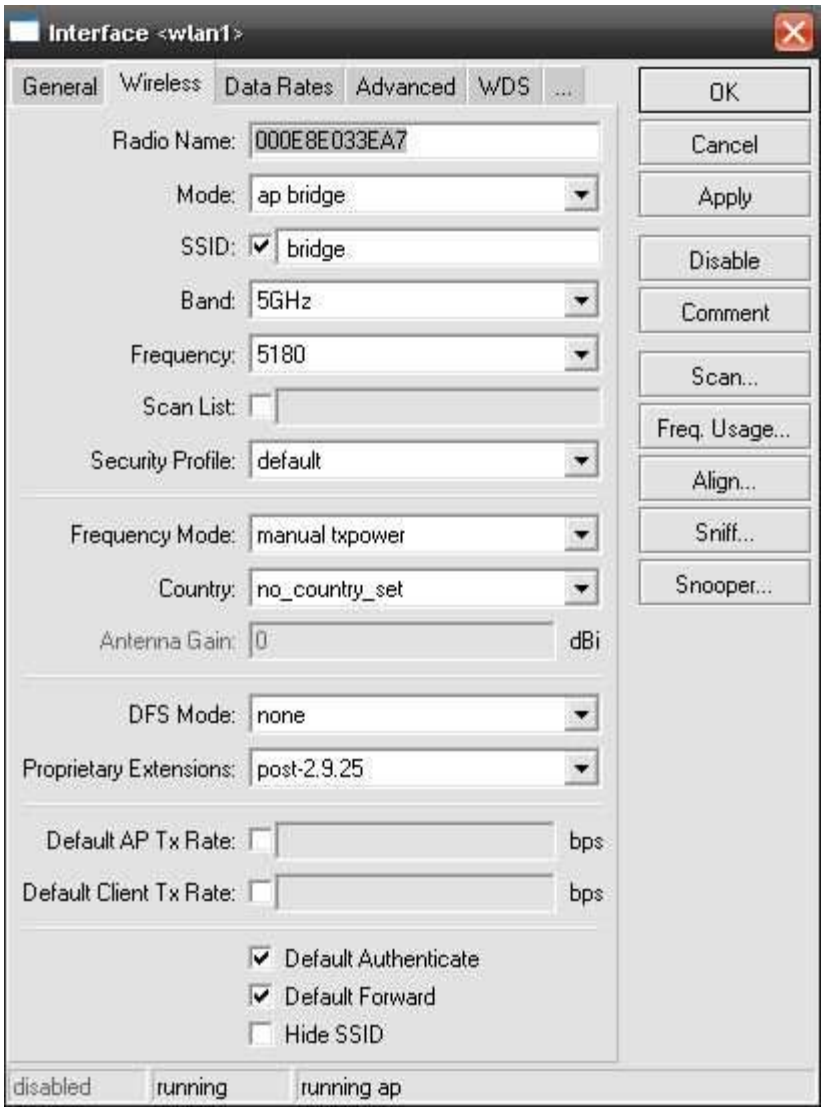
Server Setup:

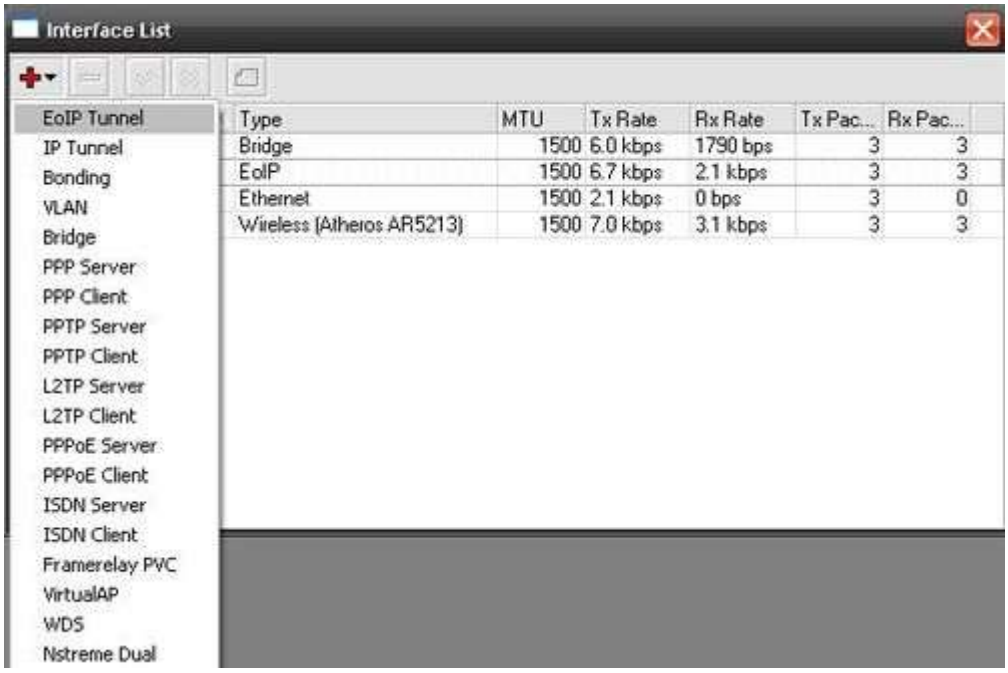
Server IP Addresses

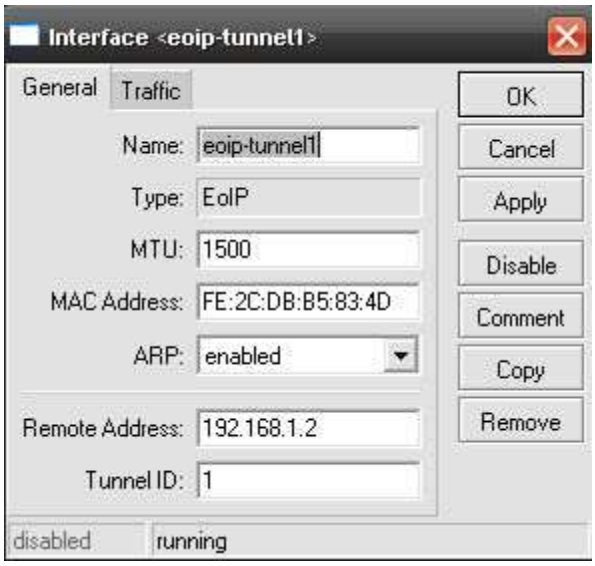
/ ip address add address=192.168.0.1/24 interface=ether1 disabled=no
/ ip address add address=192.168.1.1/24 interface=wlan1 disabled=no

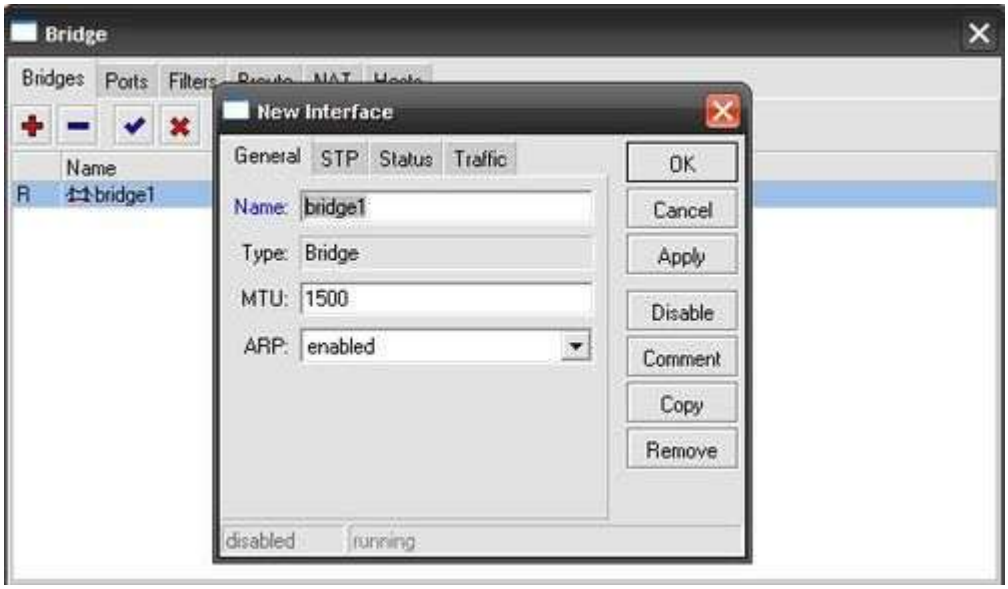

Address List
Address
Network
Broadcast
Interface
192.168.0.1/24
192.168.0.0
192.168.0.255
ether1
192.168.1.1/24
192.168.1.0
192.168.1.255
wlan1

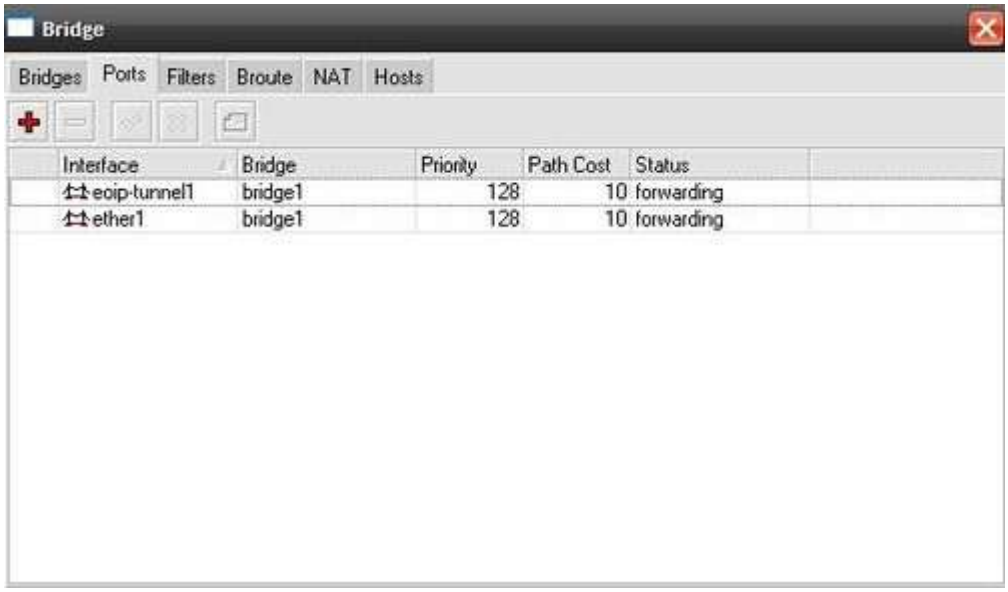

Interface List
Name
Type
MTU
Tx Rate
Rx Rate
Tx Pac...
Rx Pac...
R
bridge1
Bridge
1500
6.0 kbps
2.0 kbps
3
3
R
eoip-tunnel1
EoIP
1500
6.7 kbps
2.3 kbps
3
3
R
ether1
Ethernet
1500
2.3 kbps
0 bps
3
0
R
wlan1
1500
7.0 kbps
3.3 kbps
3
3
Add
Remove
Enable
Disable
Comment
Select All


Interface <wlan1>
General
Wireless
Data Rates
Advanced
WDS
...
OK
Cancel
Apply
Disable
Comment
Scan...
Freq. Usage...
Align...
Sniff...
Snooper...
Radio Name: 000E8E033EA7
Mode: ap bridge
SSID: bridge
Band: 5GHz
Frequency: 5180
Scan List:
Security Profile: default
Frequency Mode: manual txpower
Country: no_country_set
Antenna Gain: 0 dBi
DFS Mode: none
Proprietary Extensions: post-2.9.25
Default AP Tx Rate: bps
Default Client Tx Rate: bps
Default Authenticate
Default Forward
Hide SSID
disabled
running
running ap


Interface List
EoIP Tunnel
IP Tunnel
Bonding
VLAN
Bridge
PPP Server
PPP Client
PPTP Server
PPTP Client
L2TP Server
L2TP Client
PPPoE Server
PPPoE Client
ISDN Server
ISDN Client
Frame relay PVC
VirtualAP
WDS
Nstreme Dual
Type MTU Tx Rate Rx Rate Tx Pac... Rx Pac...
Bridge 1500 6.0 kbps 1790 bps 3 3
EoIP 1500 6.7 kbps 2.1 kbps 3 3
Ethernet 1500 2.1 kbps 0 bps 3 0
Wireless (Atheros AR5213) 1500 7.0 kbps 3.1 kbps 3 3


Interface <eoip-tunnel1>
General
Traffic
OK
Cancel
Apply
Disable
Comment
Copy
Remove
Name: eoip-tunnel1
Type: EoIP
MTU: 1500
MAC Address: FE:2C:DB:B5:83:4D
ARP: enabled
Remote Address: 192.168.1.2
Tunnel ID: 1
disabled
running

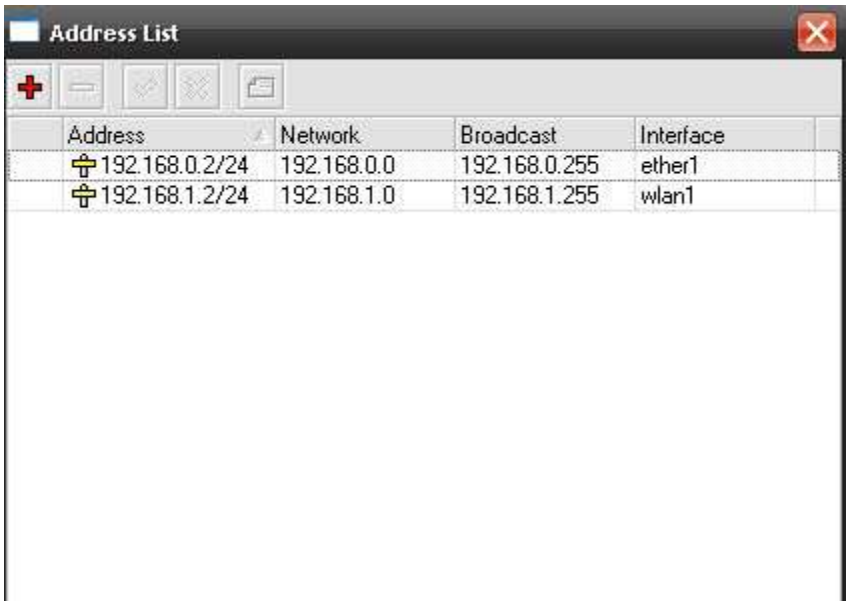


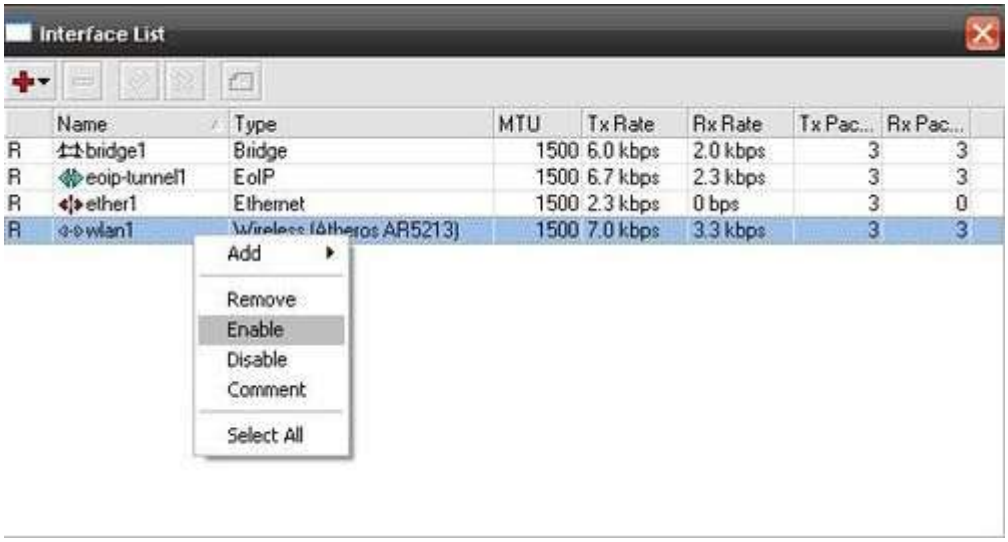


Client Setup:

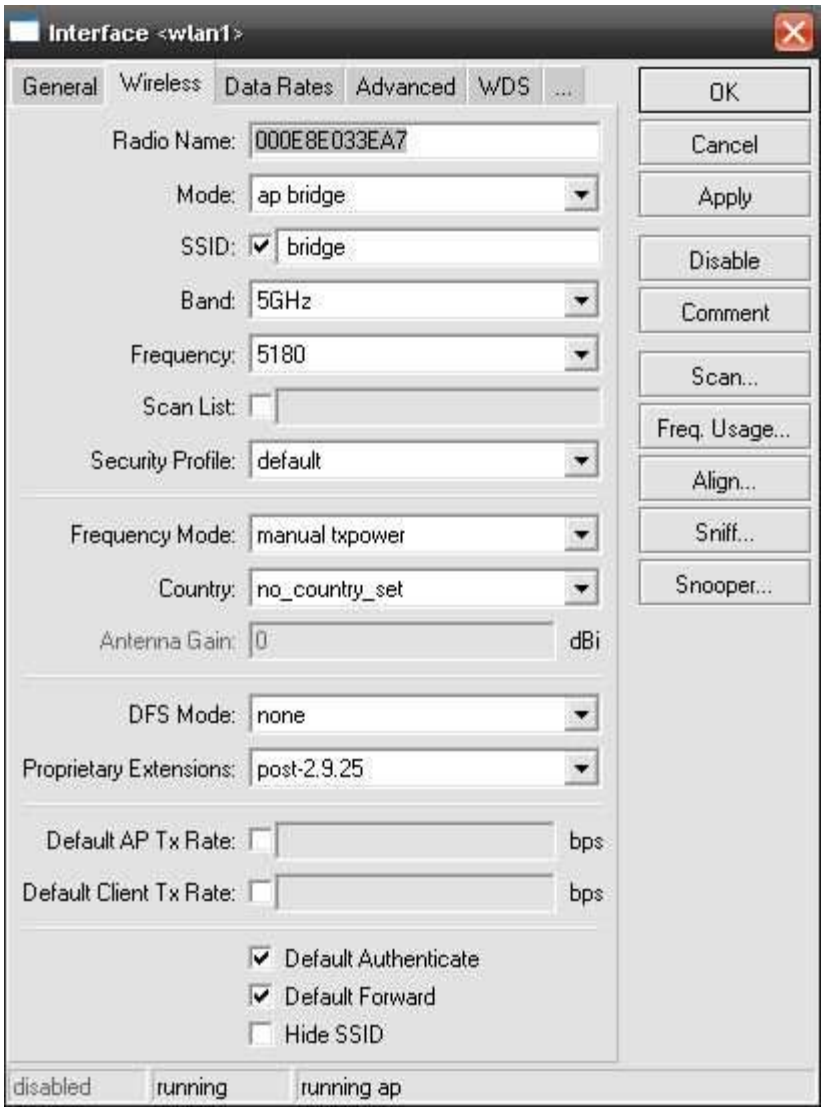
Client IP Addresses

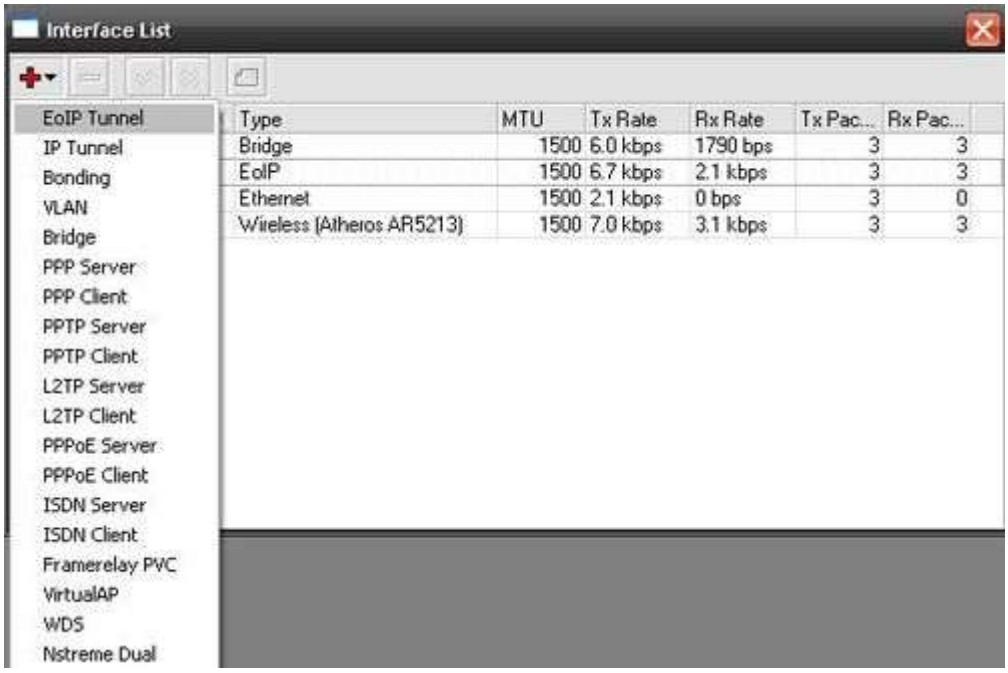
/ ip address add address=192.168.0.2/24 interface=ether1 disabled=no
/ ip address add address=192.168.1.2/24 interface=wlan1 disabled=no

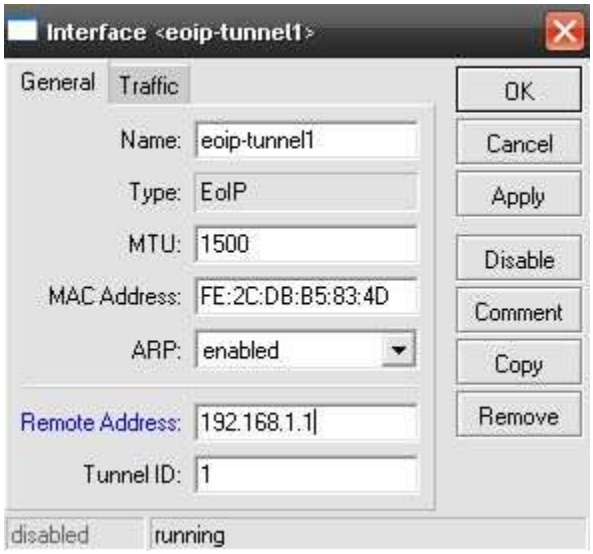


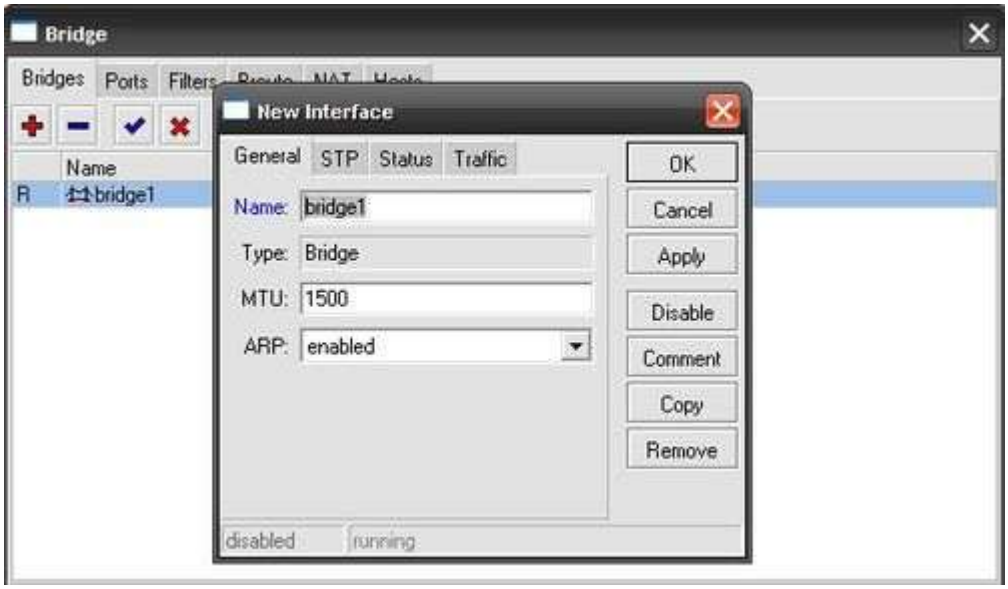


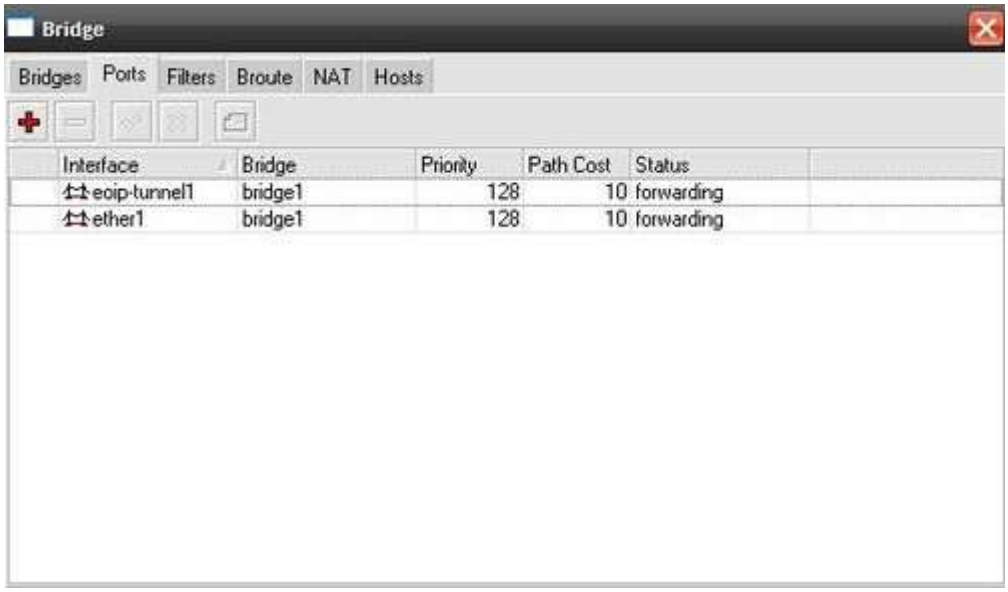
Note that this image is incorrect - Set the Mode to Station and it will work fine then


Interface <wlan1>
General
Wireless
Data Rates
Advanced
WDS
...
OK
Cancel
Apply
Disable
Comment
Scan...
Freq. Usage...
Align...
Sniff...
Snooper...
Radio Name: 000E8E033EA7
Mode: ap bridge
SSID: bridge
Band: 5GHz
Frequency: 5180
Scan List:
Security Profile: default
Frequency Mode: manual txpower
Country: no_country_set
Antenna Gain: 0 dBi
DFS Mode: none
Proprietary Extensions: post-2.9.25
Default AP Tx Rate: bps
Default Client Tx Rate: bps
Default Authenticate
Default Forward
Hide SSID
disabled
running
running ap

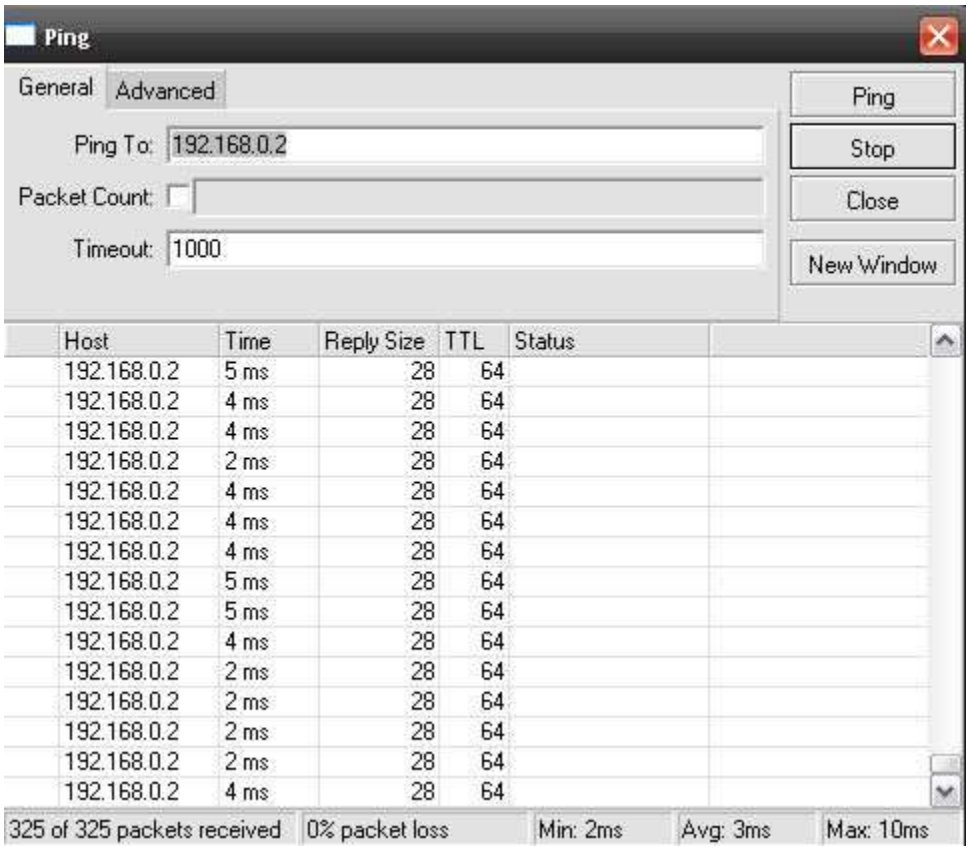

Interface List
EoIP Tunnel
IP Tunnel
Bonding
VLAN
Bridge
PPP Server
PPP Client
PPTP Server
PPTP Client
L2TP Server
L2TP Client
PPPoE Server
PPPoE Client
ISDN Server
ISDN Client
Frame relay PVC
VirtualAP
WDS
Nstreme Dual
Type MTU Tx Rate Rx Rate Tx Pac... Rx Pac...
Bridge 1500 6.0 kbps 1790 bps 3 3
EoIP 1500 6.7 kbps 2.1 kbps 3 3
Ethernet 1500 2.1 kbps 0 bps 3 0
Wireless (Atheros AR5213) 1500 7.0 kbps 3.1 kbps 3 3


Interface <eoip-tunnel1>
General
Traffic
OK
Cancel
Apply
Disable
Comment
Copy
Remove
Name: eoip-tunnel1
Type: EoIP
MTU: 1500
MAC Address: FE:2C:DB:B5:83:4D
ARP: enabled
Remote Address: 192.168.1.1
Tunnel ID: 1
disabled
running





Check your setup



Troubleshooting HR-WRAP

Introduction

This guide provides a quick reference for troubleshooting your HR-WRAP system. It is intended to familiarize you with HSConfigurator (HSConf.) so that the appropriate configuration can be restored or applied.

Your HR-WRAP comes from the factory preset with an IP address of 192.168.1.1. This addresses can be used, per the user manual, to access the device via either HTTP (web), telnet, ssh or HSConf.

The outdoor HR-WRAP does not have any reset buttons or external indicators in order to maintain it's water tight seal. All management and diagnostics are via the Ethernet port, so it is essential to restore this port to a functioning state in order to manage the system.

Typically, the errors that cause a system to become unreachable are:

- Bad power. Varying power voltages, outages, and surges can damage your unit. Make sure that you use a UPS (Un-interruptible power supply) with all power conditioning placed in front of the power supply feeding the PoE injector.
- Failed PoE injector
- Failed power transformer
- Crossed (shorted) pins in the RJ45 jacks
- Shorts or opens in the CAT5 cable running from the HR-WRAP to the PoE injector.
- Using a cross over cable instead of a straight through cable between the HR-WRAP and the PoE injector.
- Using a straight through cable instead of a cross over between a PC and the PoE.
- Using a cross over cable instead of a straight through between a Ethernet Switch and the PoE.

- Deleting the IP address of the HR-WRAP using the IP Address dialog on the HR-WRAP
- Changing the IP address to an unknown value using the IP Address dialog on the HR-WRAP
- Applying an IP address without the /24 subnet. Example: 192.168.1.1/24
- Applying firewall rules to the HR-WRAP which filter or limit access to the device for management through the Ethernet port (most common mistake is to apply the Hotspot setup wizard to the Ether1 port)
- Turning off the telnet, ssh or web interfaces on the HR-WRAP by disabling the services.
- Failed processor board in the HR-WRAP

The typical steps to repair the system are:

Check for power issues

39 Use a UPS which provides stable power with no voltage sags or surges.

40 Make sure that you are using the supply provided with the HR-WRAP. This is a 18VDC supply or a 12VDC supply (optional). HauteSpot also provides 15VDC supplies.

41 Check the green power indicator on the PoE injector. The light should be on solid green at all times. If this light goes out when Ethernet cables are attached, you have a short between pins 4/5 and 7/8 or a loose power connection.

42 Remove the top cover of the HR-WRAP using an 8mm socket wrench (voids one year warranty). Connect an Ethernet cable between the HR-WRAP and the PoE injector. Located between the Ethernet connector and the serial connector on the processor board are three small LEDs. The one closest to the Ethernet connector should be on solid green. This means that the processor board is receiving good power.

43 If the green LED is not on, then your board is not receiving power. To verify that the board is functioning properly:

44 Inside the case of the HR-WRAP, disconnect the RJ45 ethernet cable jumper from the jack on board the processor board. This is a

grey/black cable that runs from the water tight RJ45 to the RJ45 on the board.

45 Connect the power connector from the wall power supply provided with the HR-WRAP directly to the black connector on the processor board, located immediately adjacent to the Ethernet connector. The green LED should now go on.

46 If the LED goes on, then:

47 Check the RJ45 connector on the processor board. Look at the pins in the connector to make sure that none are bent or touching.

48 Directly connect your Ethernet cable from your PoE to the RJ45 connector on the processor board. Re-attach the power supply to the PoE. If the LED goes on, then the connection between the waterproof connector and the processor board has failed. Contact HauteSpot Networks Technical support for replacement.

Correcting cabling errors:

49 Check the termination of your RJ-45 ice cube connectors to make sure that they are straight through (WhOr|OrWh|WhGr|BlWh|WhBl|GrWh|WhBr|BrWh). If possible, test the cable using a pair scanner or by plugging into a hub or switch and then attaching a computer at the far end. Refer to the user manual for specific wiring guidelines and pin out configurations

50 Make sure that you are using straight through Ethernet cables between your Power Over Ethernet injector and the HR-WRAP

51 For use with a computer (direct connection) use a cross over cable (provided with system) between the PoE injector and your computer.

52 For use with a Ethernet switch (network connection) use a straight through cable between the PoE injector and your switch

Correcting IP numbering errors:

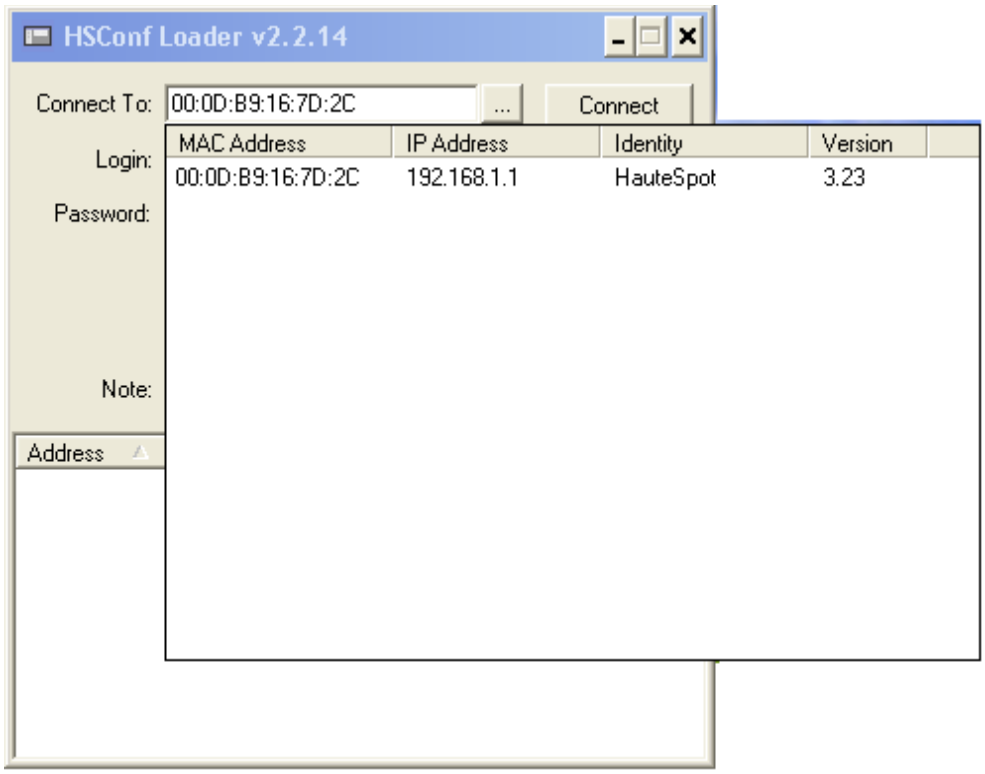
53 First, attach your HR-WRAP directly to a PC using MS Windows XP Windows 2000 or Windows Vista by using a cross over cable between your PC and the PoE injector and a straight through cable between the PoE injector and the HR-WRAP

54 Next, configure your PC to use the IP address 192.168.1.10 with a subnet mask of 255.255.255.0. Do not define a default gateway or DNS server. Refer to the 'Static IP on a PC' document for details.

55 Open a command prompt on your computer and then type “ping 192.168.1.1”. If you get a return packet response, then your system is running and addressed per the factory default. If you do not get a response, proceed to the next step.

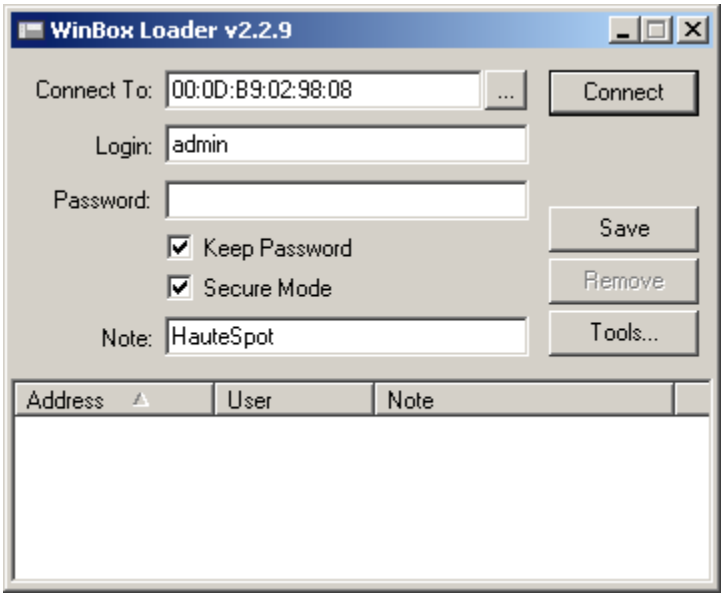
56 Download a copy of HSConf. loader from the HauteSpot Networks by loggin into our support page and downloading it here: http://www.hautespot.net/cgi/download.php?f=HSConf.exe

57 Now open the HSConf. loader.

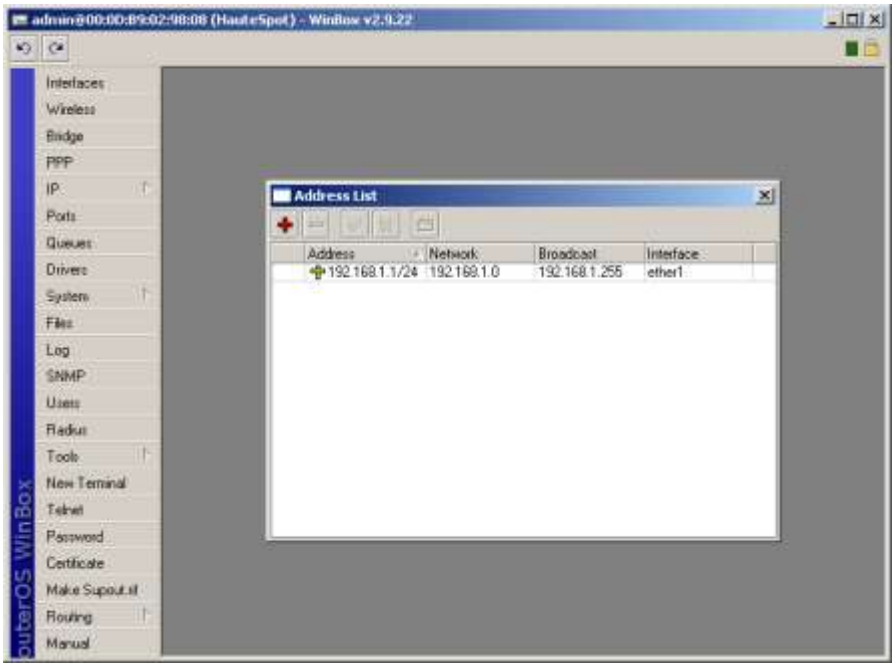


58 Next to the 'Connect To:' box you will see a box with three dots "…" Click on this box and a drop down list will appear. This list uses MAC layer discovery to find every local HR-WRAP device. If your device is still functioning, then there should be an entry for it on this list.

Click on the line with the unit you want to connect to. Then enter the default Login ID of admin with a blank password and click the connect button:



This should open up the HSConf. interface and you should then be able to update your configuration with proper IP addressing using the IP Address dialog.

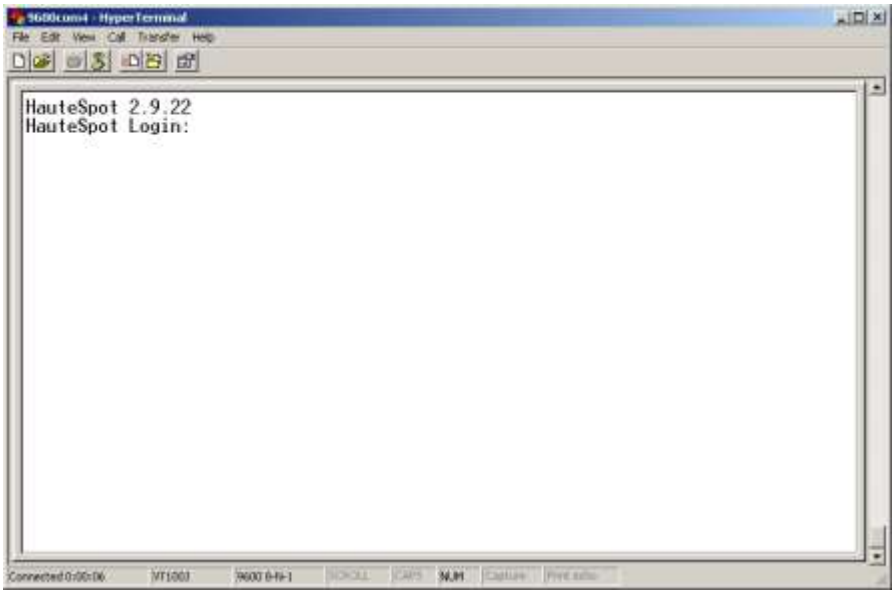


Set the IP Address back to 192.168.1.1. Now attempt to ping the unit again from your PC and connect using a web browser.

If you cannot connect via HSConf., then you will need to completely reset the HR-WRAP.

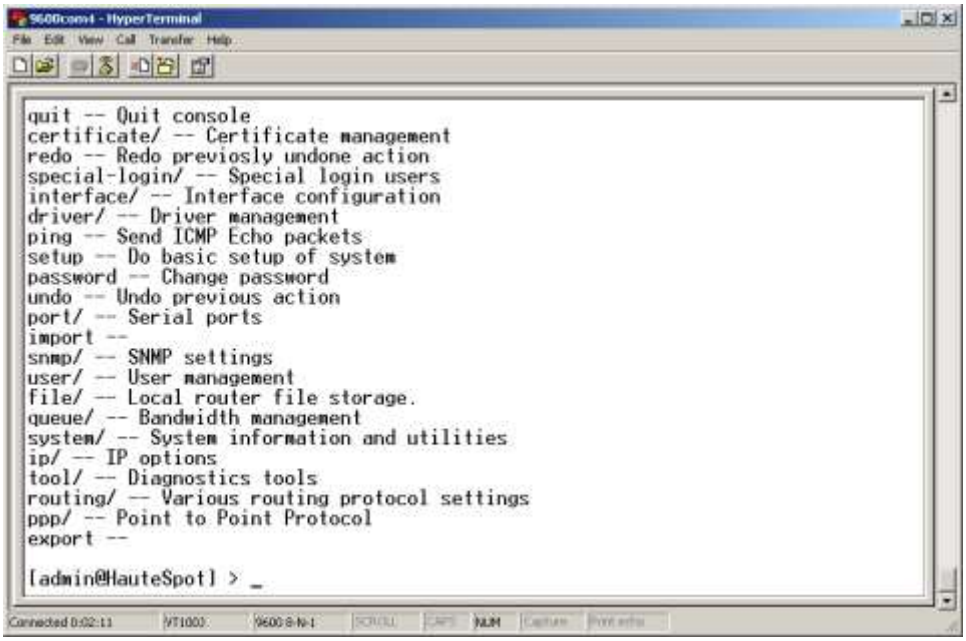
To do this:

59 Remove the cover of the unit using a 8mm wrench (voids one year warranty).
60 Make sure that power is on and that the green LED next to the Ethernet port on the processor board is lit solid green.
61 Using a DB-9 RS-232 Null Modem cable, attach your HR-WRAP to the COM port of your computer.
62 Using Hyperterm, open a session connected to the COM port with the following configuration parameters:
    63 115200 baud (9600 for older devices)
    64 8 data bits
    65 No parity bits
    66 1 stop bit
    67 No flow control
68 This should open a console session to the HR-WRAP that looks like:



You can log in using the Login ID of admin with no password.

69 You can get a menu of available command options by typing the “?” key:


quit -- Quit console
certificate/ -- Certificate management
redo -- Redo previosly undone action
special-login/ -- Special login users
interface/ -- Interface configuration
driver/ -- Driver management
ping -- Send ICMP Echo packets
setup -- Do basic setup of system
password -- Change password
undo -- Undo previous action
port/ -- Serial ports
import --
snmp/ -- SNMP settings
user/ -- User management
file/ -- Local router file storage.
queue/ -- Bandwidth management
system/ -- System information and utilities
ip/ -- IP options
tool/ -- Diagnostics tools
routing/ -- Various routing protocol settings
ppp/ -- Point to Point Protocol
export --
[admin@HauteSpot] > _

70 To reset your unit completely, enter the command “system reset-configuration”. This will completely erase all of your router’s configuration details, including IP addressing.

Your HR-WRAP is now configured to the default address and you should be able to reconnect to it using the instructions in step 3 above.

If you are still not able to access your HR-WRAP system, please contact technical support at 805-541-9477 for more assistance.

HauteRouterOS UPnP

Document revision: 2.2 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

The HauteRouterOS supports Universal Plug and Play architecture for transparent peer-to-peer network connectivity of personal computers and network-enabled intelligent devices or appliances. UPnP builds enables these devices to automatically connect with one another and work together to make networking possible for more people.

Specifications

Packages required: ***system***
License required: *Level1*
Submenu level: ***/ip upnp***
Standards and Technologies: TCP/IP, HTTP, XML, IGD
Hardware usage: *Not significant*

Description

UPnP enables data communication between any two devices under the command of any control device on the network. Universal Plug and Play is completely independent of any particular physical medium. It supports networking with automatic discovery without any initial configuration, whereby a device can dynamically join a network. DHCP and DNS servers are optional and will be used if available on the network. UPnP implements simple yet powerfull NAT traversal solution, that enables the client to get full peer-to-peer network support from behind the NAT.

There are two interface types for UPnP: internal (the one local clients are connected to) and external (the one the Internet is connected to). A router may only have one external interface with a 'public' IP address on it, and as many internal IP addresses as needed, all with source-NATted 'internal' IP addresses.

The UPnP protocol is used for most of DirectX games as well as for various Windows Messenger features (remote asisstance, application sharing, file transfer, voice, video) from behind a firewall.

Additional Resources

UPnP Forum

Enabling Universal Plug-n-Play

Submenu level: ***/ip upnp***

Property Description

allow-disable-external-interface (yes | no; default: **yes**) - whether or not should the users be allowed to disable router's external interface. This functionality (for users to be able to turn the router's external interface off without any authentication procedure) is required by the standard, but as it is sometimes not expected or unwanted in UPnP deployments which the standard was not designed for (it was designed mostly for home users to establish their local networks), you can disable this behavior
enabled (yes | no; default: **no**) - whether UPnP feature is enabled
show-dummy-rule (yes | no; default: **yes**) - this is to enable a workaround for some broken implementations, which are handling the absense of UPnP rules inincorrectly (for example, popping up error messages). This option will instruct the server to install a dummy (meaningless) UPnP rule that can be observed by the clients, which refuse to work correctly otherwise

Example

To enable UPnP feature:

```
[admin@Hautespot] ip upnp> set enable=yes
[admin@Hautespot] ip upnp> print
                           enabled: yes
    allow-disable-external-interface: yes
                     show-dummy-rule: yes
[admin@Hautespot] ip upnp>
```

UPnP Interfaces

Submenu level: ***/ip upnp interfaces***

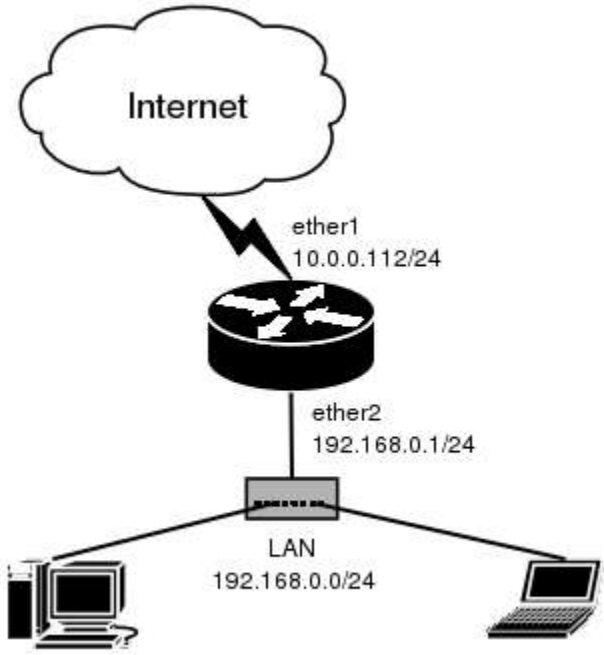
Property Description

interface (*name*) - interface name UPnP will be run on
type (external | internal) - interface type, one of the:
external - the interface global IP address is assigned to
internal - router's local interface

Notes

It is highly recommended to upgrade DirectX runtime libraries to version DirectX 9.0c or higher and Windows Messenger to versionWindows Messenger 5.0 or higher in order to get UPnP to work properly.

Example



We have masquerading already enabled on our router:

```
[admin@Hautespot] ip upnp interfaces> /ip firewall src-nat print
Flags: X - disabled, I - invalid, D - dynamic
  0   src-address=0.0.0.0/0:0-65535 dst-address=0.0.0.0/0:0-65535
      out-interface=ether1 protocol=all icmp-options=any:any flow=""
      connection="" content="" limit-count=0 limit-burst=0 limit-time=0s
      action=masquerade to-src-address=0.0.0.0 to-src-port=0-65535

[admin@Hautespot] ip upnp interfaces>
```

Now all we have to do is to add interfaces and enable UPnP:

```
[admin@Hautespot] ip upnp interfaces> add interface=ether1 type=external
[admin@Hautespot] ip upnp interfaces> add interface=ether2 type=internal
[admin@Hautespot] ip upnp interfaces> print
Flags: X - disabled
  #   INTERFACE TYPE
  0 X ether1    external
  1 X ether2    internal
```

```
[admin@Hautespot] ip upnp interfaces> enable 0,1
[admin@Hautespot] ip upnp interfaces> .. set enabled=yes
[admin@Hautespot] ip upnp interfaces>
```

HauteRouterOS User Manager

Document revision: 2.2 (January 2, 2009, 10:45 GMT)
Applies to: V3.0

General Information

Summary

User Manager is a RADIUS authentication database and application which runs under HauteRouterOS and can provide access control, billing, etc for any wired or wireless network. It can provide all of the access control features required by any network and is extremely secure.

Getting Started

Create first subscriber

Note: Starting from version 3.0 a default subscriber with login admin and empty password is created when User Manager package is installed for the first time. I.e., admin subscriber is created only if the User Manager package was not installed prior to version 3.0.

First subscriber must be added using HauteRouterOS terminal (console). All the configuration is done under the /tool user-manager menu.

To create a subscriber you should go to /tool user-manager customer menu and execute add command. It will ask for the username which you will use.

or you can enter this into the command line:

```
[admin@USER_MAN] tool user-manager customer> add login="admin"
password="adminpassword" permissions=owner
```

After that you can use print command to see what you have added and you can also set the password.

```
[admin@USER_MAN] tool user-manager customer> print
  Flags: X - disabled
   0   subscriber=admin login="admin" password="adminpassword" time-
zone= 00:00 permissions=owner parent=admin
```

After that you can use the web interface.
Use web interface

To log on customer web interface type the following address in your web browser:
http://Router_IP_address/userman

where "Router_IP_address" must be replaced with IP address of your router.

Use login and password of the subscriber you have created in console.

Wireless Hotspot Configuration Introduction

To make this setup, you should have running Hotspot server on the router. Let us consider configuration steps for HotSpot and User Manager routers, in order to use User Manager for HotSpot users.

HotSpot configuration

* Set HotSpot to use User Manager for HotSpot server users,

```
/ ip hotspot profile set hsprof1 use-radius=yes
```

* Add radius client to consult User Manager for HotSpot service.

```
/ radius add service=hotspot address=y.y.y.y secret=123456
```

'secret' is equal to User Manager router secret. 'y.y.y.y' is the User Manager router address. By default this is 127.0.0.1. If using a remotely located Router (perhaps via a VPN) then the IP address entered is the IP address of that remote Router. The router could be a Radius Server, or another ROS with User Manager installed.

* Note, first local HotSpot database is consulted, then User Manager database.

It means that if you have configuration in '/ ip hotspot user print', users will be able to authenticate in HotSpot using these data. Delete users configuration from '/ ip hotspot print' to stop using local HotSpot database for authentication. To move batch of local HotSpot users to the User Manager database use export/import . Use text editor program to create appropriate file to import local users to the User Manager database.

User Manager configuration

* First, you need to download and install User Manager package;
* Create User Manager subscriber;

```
/ tool user-manager customer add login="HauteSpot" password="qwerty"
permissions=owner
```

* Add HotSpot router information to router list,

```
/ tool user-manager router add subscriber=HauteSpot ip-address=x.x.x.x
shared-secret=123456
```

'x.x.x.x' is the address of the HotSpot router, 'shared-secret' should match on both User Manager and HotSpot routers. Adding 'x.x.x.x' as a router allows Radius requests from 'x.x.x.x' to be passed to the Radius Server built into User Manager. Therefore if you have any remote ROS

Hotspots that require access to this Radius Server, then all their IP addresses must be added to this list.

* Add HotSpot user information, it is equal to 'ip hotspot user' when local HotSpot is used for clients

```
 / tool user-manager user add username=demo password=demo
subscriber=HauteSpot
```

We discuss only basic configuration example, detailed information about 'user' menu configuration.

* You can use User Manager web interface after first subscriber created.

* To make sure, that client is using User Manager for AAA,

```
 / ip hotspot active print
 Flags: R - radius, B - blocked
  #    USER          ADDRESS         UPTIME       SESSION-TIME-LEFT
IDLE-TIMEOUT
  0 R  00:01:29:2... 192.168.100.2   1m29s
```

'R' means that client uses User Manager server for AAA services.

User Manager/PPP Introduction

User Manager can be used as a remote authentication, authorization and accounting server for PPP clients.

Since 2.9.35 PAP,CHAP, MS-CHAPv1 and MS-CHAPv2 protocols are supported by the User Manager.

Let us consider the following configuration steps for PPP and User Manager routers.

PPP configuration

We consider PPPoE server <-> PPPoE client configuration example, where the PPPoE server uses a remote User Manager database for PPPoE client authentication, authorization and accounting. Both PPPoE server and PPPoE client are HauteSpot routers, any other PPPoE client might be used instead.

PPP server configuration

* First, add the PPPoE server to the local interface, :

```
 / interface pppoe-server server add interface=ether1 service-
name=HauteSpot one-session-per-host=yes disabled=no
```

* Specify the use of User Manager for PPPoE clients:

```
 / ppp aaa set use-radius=yes
```

* Set IP address of the PPPoE server, IP address might not be assigned to the interface of PPPoE server. Moreover static IP address or DHCP should not be used on the same interfaces as the PPPoE server for security reasons.

```
/ ppp profile set default local-address=192.168.0.1
```

* Add radius client to consult User Manager for PPP service.

```
/ radius add service=ppp address=y.y.y.y secret=123456
```

'secret' is equal to User Manager router secret. 'y.y.y.y' is the User Manager router address.

* Note, first the local PPP database is consulted, then the User Manager database.

PPP client configuration

* Add PPPoE client to the interface

```
/ interface pppoe-client add interface=ether1 user=HauteSpot
password=HauteSpot service-name=HauteSpot disabled=no
```

User Manager configuration

* First, you need to download and install User Manager package;
* Create User Manager subscriber;

```
/ tool user-manager customer add login="HauteSpot" password="qwerty"
permissions=owner
```

* Add PPP server information to router list,

```
/ tool user-manager router add subscriber=HauteSpot ip-address=x.x.x.x
shared-secret=123456
```

'x.x.x.x' is the address of the PPPoE-server router, 'shared-secret' should match on both User Manager and PPPoE-server routers.

* Add PPPoE client information,

```
/ tool user-manager user add username=demo password=demo
subscriber=HauteSpot ip-address=192.168.0.2
```

* Let us verify, that PPPoE client is connected and using User Manager for authentication, authorization and accounting. First we monitor if PPPoE client is connected, then we verify that User Manager was used. The first command is executed on PPPoE client router, second on PPPoE server:

```
/ interface pppoe-client monitor pppoe-out1
      status: "connected"
      uptime: 12h2m29s
   idle-time: 12h2m17s
service-name: "HauteSpot"
     ac-name: "HauteSpot"
      ac-mac: 00:0C:42:05:54:8F
```

```
          mtu: 1480
          mru: 1480

 / ppp active> print
 Flags: R - radius
  #   NAME         SERVICE CALLER-ID         ADDRESS         UPTIME
ENCODING
  0 R HauteSpot     pppoe   00:0C:42:05:54:6E 192.168.0.2     12h1m48s
```

User Manager/DHCP Introduction

To make this setup, you should have running DHCP server on the router. Let's consider configuration steps for DHCP and User Manager routers, in order to use User Manager for DHCP server users.

DHCP router configuration

* Set DHCP to use User Manager for DHCP server leases,

```
/ ip dhcp-server set dhcp1 use-radius=yes
```

* Add radius client to consult User Manager for DHCP service.

```
/ radius add service=dhcp address=y.y.y.y secret=123456
```

'secret' is equal to User Manager router secret. 'y.y.y.y' is the User Manager router address.

* Note, first local router database is consulted, then User Manager database. User will be unable to obtain DHCP lease, if DHCP router and User Manager server will not contain any information about user's data.

User Manager configuration

* First, you need to download and install User Manager package;

* Create User Manager subscriber;

```
/ tool user-manager customer add login="HauteSpot" password="qwerty"
permissions=owner
```

* Add DHCP router information to router list,

```
/ tool user-manager router add subscriber=HauteSpot ip-address=x.x.x.x
shared-secret=123456
```

'x.x.x.x' is the address of the DHCP router, 'shared-secret' should match on both User Manager and DHCP routers.

* Add DHCP user information, that client with MAC address 00:01:29:27:81:95 will always receive 192.168.100.2 address. User will receive dynamic address from the DHCP ip pool, if ip-address is not specified.

```
/ tool user-manager user add add subscriber=HauteSpot
username="00:01:29:27:81:95" ip-address=192.168.100.2
```

We discuss only basic configuration example, detailed information about user menu configuration.

* To make sure, that user is receiving lease from User Manager,

```
 / ip dhcp-server lease> print
 Flags: X - disabled, R - radius, D - dynamic, B - blocked
 #   ADDRESS         MAC-ADDRESS       HOST-NAME    SERVER RATE-LIMIT
STATUS
 0 R 192.168.100.2   00:01:29:27:81:95              dhcp1
bound
```

'R' means that lease has been received from User Manager server.

User Manager/Wireless Introduction

We consider the scenario for wireless network, when only clients from User Manager database are able to establish communications with 'Access Point' router. To make this setup, you must have running Access Point. Let us consider configuration steps for Access Point and User Manager routers.

Access Point configuration

* Set Access Point to use User Manager for wireless client authentication,

```
 / interface wireless security-profiles set default radius-mac-
authentication=yes
```

* Add radius client to consult User Manager for wireless service.

```
/ radius add service=wireless address=y.y.y.y secret=123456
```

'secret' is equal to User Manager router secret. 'y.y.y.y' is the User Manager router address.

* Note, first local router database is consulted, then User Manager database. Wireless client will be unable to connect to Access Point, if Access Points router does not contain any entry in the 'interface wireless access-list' for the particular configuration and User Manager server will not have any information about user's data.

* Make sure you do not have any entry in the 'interface wireless access-list', remove all hosts from 'access-list' to ensure wireless client MAC authentication only via User Manager,

```
 / interface wireless access-list remove [find]
```

User Manager configuration

* First, you need to download and install User Manager package;

* Create User Manager subscriber;

```
/ tool user-manager customer add login="HauteSpot" password="qwerty"
permissions=owner
```

* Add Access Point router information to router list,

```
/ tool user-manager router add subscriber=HauteSpot ip-address=x.x.x.x
shared-secret=123456
```

'x.x.x.x' is the address of the Access Point router, 'shared-secret' must match on both User Manager and Access Point routers.

* Add wireless client information, client MAC-address that is allowed to establish connection to the Access Point,

```
/ tool user-manager user add subscriber=HauteSpot
username="00:01:29:27:81:95"
```

User Manager/HauteRouterOS user Introduction

User Manager server might be used as remote storage of HauteRouterOS login and password information. HauteSpot router will consult User Manager for login and password, when you are accessing HauteRouterOS via HSConfig or console session. Let us consider configuration steps. RouterOS configuration

* Set HauteRouterOS to use User Manager server for checking login and password information,

```
 / user aaa set use-radius=yes
```

* '/user aaa' has 'default-group' option, that define type of the default group. Default is read permissions, if you need to allow full permissions for users stored in User Manager database

```
 / user aaa set default-group=full
```

* Add radius client to consult User Manager for login service.

```
/ radius add service=login address=y.y.y.y secret=123456
```

'secret' is equal to User Manager router secret. 'y.y.y.y' is the User Manager router address.

* Note, first local router database is consulted, then User Manager database.

User Manager configuration

* First, you need to download and install User Manager package;

* Create User Manager subscriber;

```
/ tool user-manager customer add login="HauteSpot" password="qwerty"
permissions=owner
```

* Add HauteRouterOS router information to router list,

```
/ tool user-manager router add subscriber=HauteSpot ip-address=x.x.x.x
shared-secret=123456
```

'x.x.x.x' is the address of the HauteRouterOS router, 'shared-secret' must match on both User Manager and HauteRouterOS routers.

* Add login/password information, that account will be able to access HauteRouterOS. login is HauteSpot, password is HauteSpot.

```
/ tool user-manager user add subscriber=HauteSpot username=HauteSpot
password=HauteSpot
```

User Manager/Customer page

Setup

There are no special setup actions for web interface. The only requirement - at least one subscriber must be defined. See first subscriber setup guide.

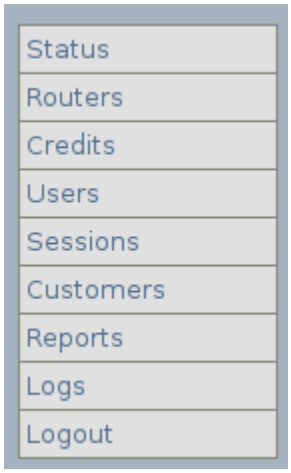
How to find?

Type the following address in your web browser: http://Router_IP_address/userman

where "Router_IP_address" must be replaced with IP address of your router.

Sections

Here are described customer page sections. Use menu on the left side to navigate:



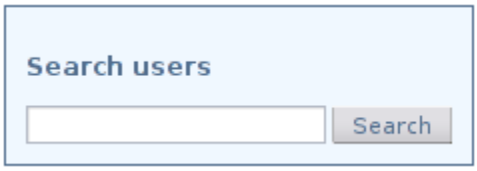
Status

This page has several components:

- User search;
- Active user listing;
- Active session listing;
- User batch-add form.

User search

Type in the search pattern and press the button "Search". Results will be displayed in a new table.

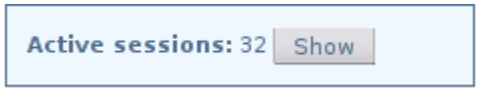


Active users

Active user count displayed here. To see a full list of active users, click on "Show":

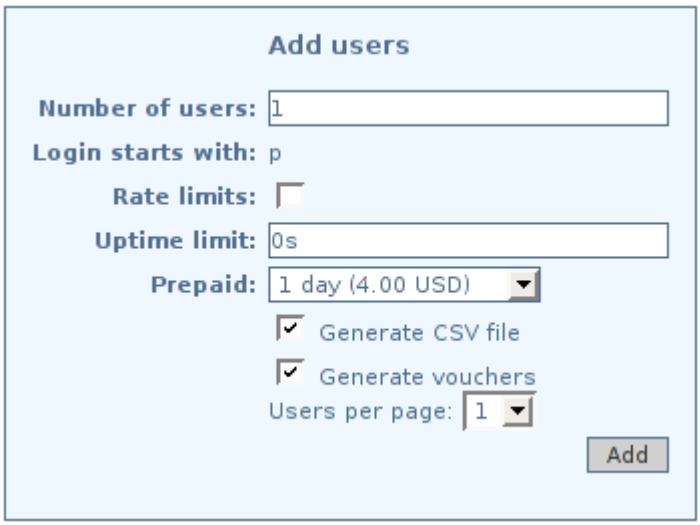
Active sessions

Active sessions count displayed here. To see a full list of active sessions, click on "Show":



User batch-add form

Batch of users can be added here:



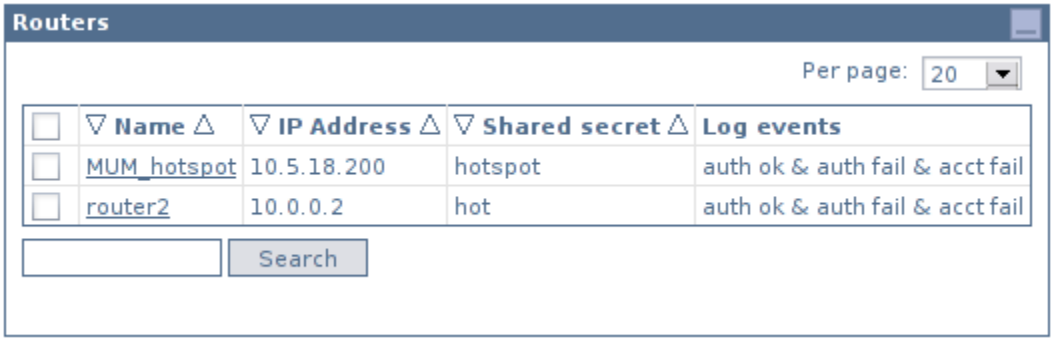
Fields:

- Number of users. How many users to add;
- Login starts with. Displays user prefix;
- Rate limits. hidden by default. Check the box on the right to show rate limit field group;
- Uptime limit;
- Prepaid. Credit that will be assigned to users. Unlimited users can also be created by selecting *unlimited* as a value.
- Generate CSV file. When checked a CSV-file will be generated containing just created user data;
- Generate vouchers. When checked printable vouchers for just created users will be generated.

Routers

View routers

Table displaying routers:

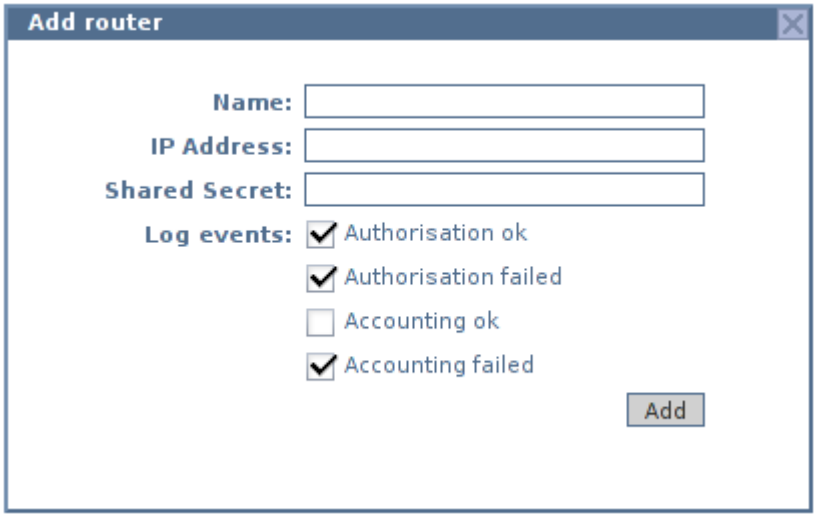


All router's attributes are shown here.

Click on *name* opens router detail/edit form.

Add router

Opens router add form. The same form is used to edit routers:



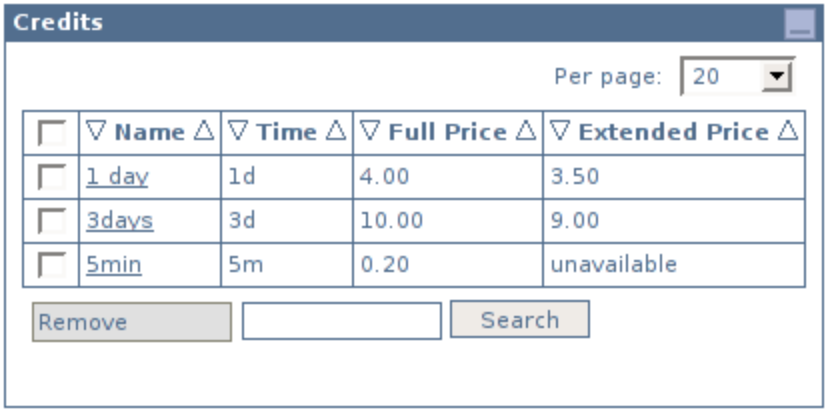
Fields:

- Name. Router's name. Must be unique per subscriber;
- IP Address. Address of the router;
- Shared secret. Password used for authentication;
- Log events. Specifies which events must be written to log.

Credits

View credits

Table displaying credits:



All credit's attributes are shown here.

Click on *name* opens credit detail/edit form.

Add credit

Opens credit add form. The same form is used to edit credits:

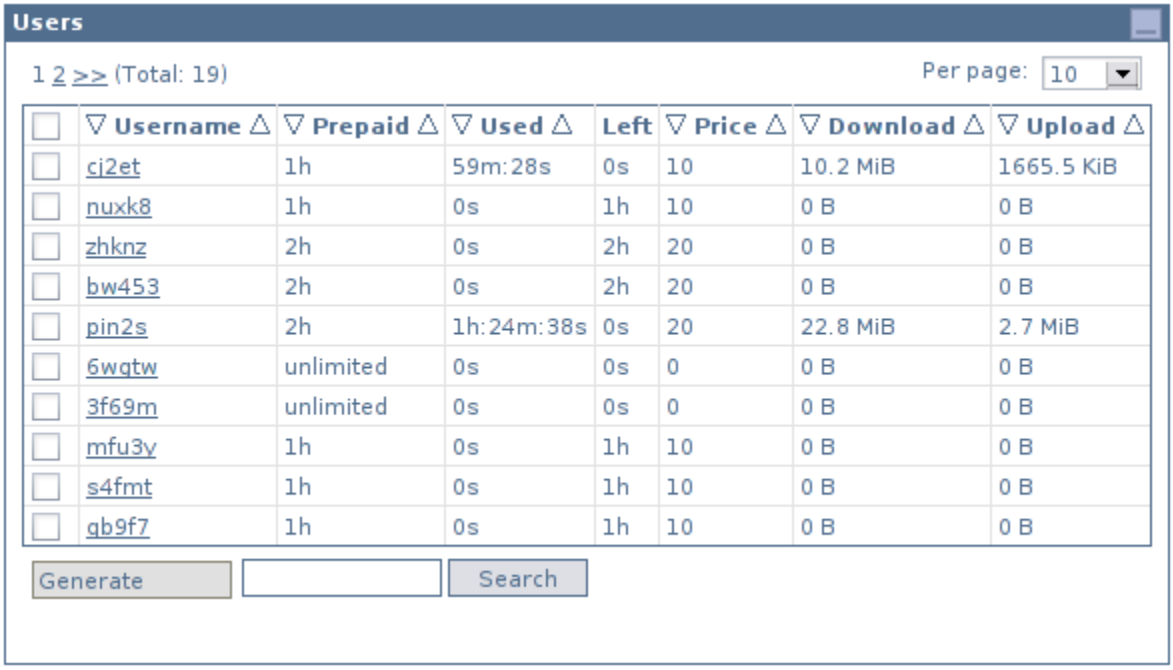
Fields:

- Name. Credit's name. Must be unique per subscriber;
- Time. How long this credit is valid when started;
- Full price. The price of this as the first credit for a user. When the checkbox at the right is empty, full price is unavailable - this credit can not be used as a base credit;
- Extended price. The price of this as extended credit for a user (user already has credits before this on). When the checkbox at the right is empty, extended price is unavailable - this credit can not be used as an extended credit;

Users

View users

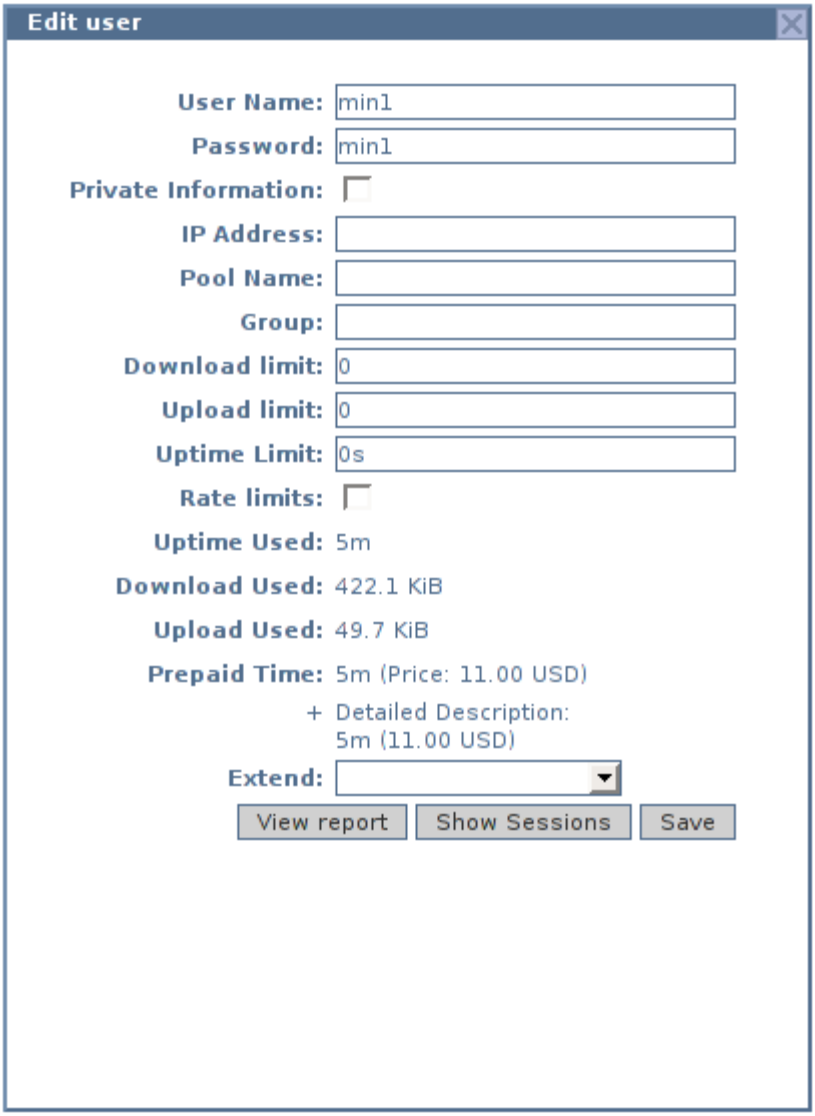
Table displaying users:


Users
1 2 >> (Total: 19)
Per page: 10
Username
Prepaid
Used
Left
Price
Download
Upload
cj2et 1h 59m:28s 0s 10 10.2 MiB 1665.5 KiB
nuxk8 1h 0s 1h 10 0 B 0 B
zhknz 2h 0s 2h 20 0 B 0 B
bw453 2h 0s 2h 20 0 B 0 B
pin2s 2h 1h:24m:38s 0s 20 22.8 MiB 2.7 MiB
6wqtw unlimited 0s 0s 0 0 B 0 B
3f69m unlimited 0s 0s 0 0 B 0 B
mfu3y 1h 0s 1h 10 0 B 0 B
s4fmt 1h 0s 1h 10 0 B 0 B
qb9f7 1h 0s 1h 10 0 B 0 B
Generate
Search

Only part of user's attributes are shown here. To see all details of specific user, open user detail form by clicking on username in the table.

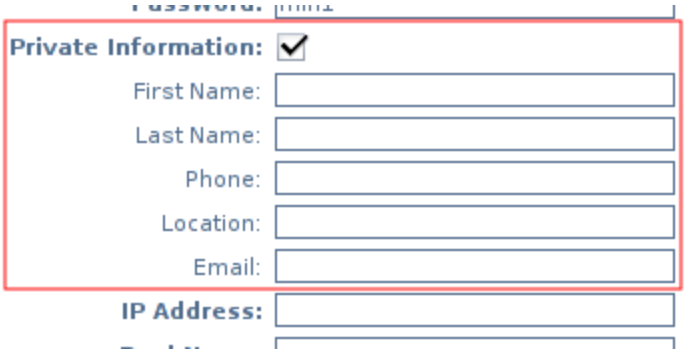
User detail form

Detail form with user data:



Contains all user fields .

There are groups of fields (for example, private information, rate limits). These fields are hidden by default and are accessible by checking the box on the right:



If the user has credits assigned the total prepaid time is shown at the bottom. To see credit details click on the plus sign ("+") under *Prepaid time*:

New credits can also be assigned (if permitted) to user. At the bottom is a select-box called "Extend" (called "Add time" when user has no credits yet). The price depends on what kind of credit this is for a user - first or extended. Price is shown in braces:

.

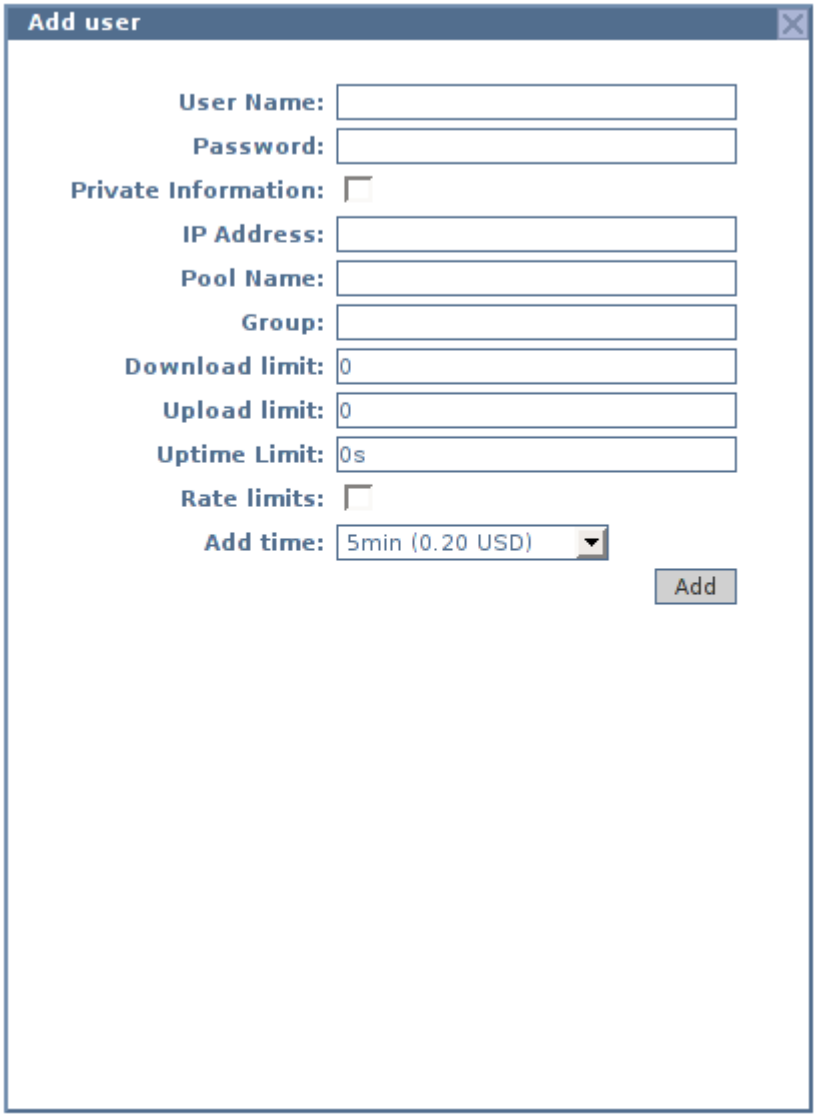
To assign credit to the user, choose the desired credit and click *Save*.

Options (buttons at the bottom):

- Save - saves edited information, assigns credit, if one selected;
- View report - opens single user report.
- Remove last credit - removes last credit that's not started yet;
- Show sessions - opens window with all sessions this user has;

Add user

Detail form for filling in information about the new user. Very similar to user detail form. This form does not have read-only counters and other user statistics:



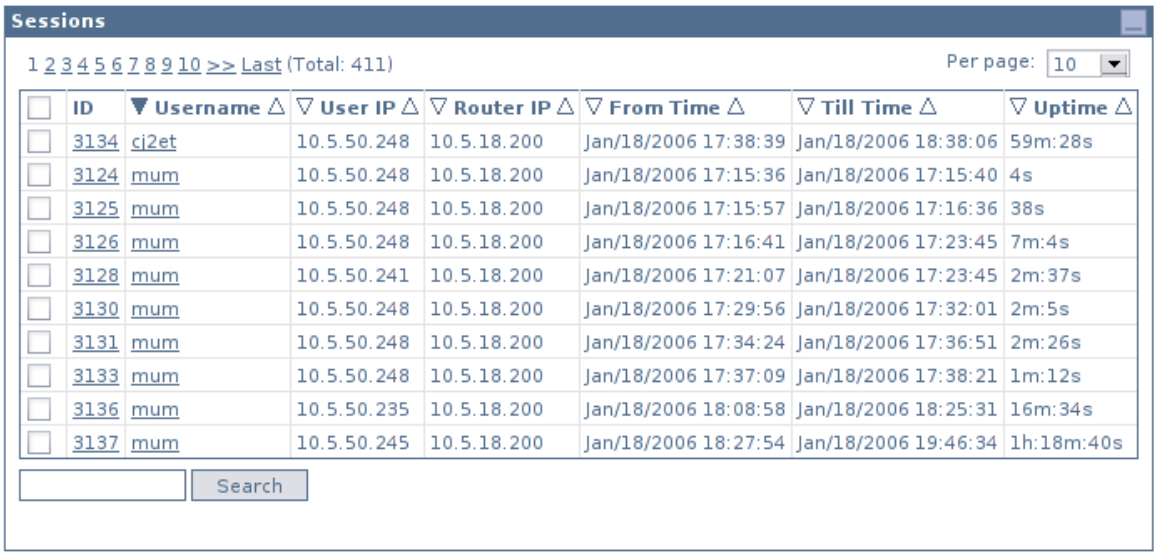
Add batch of users

The User batch-add form will be opened.

Sessions

View sessions

Table displaying sessions:



Only part of session's attributes are shown here. To see all details of specific session, open session detail form by clicking on ID in the table.

To see details of session user click on the username in the table.

Session detail form

Detail form with session data:

Contains all session fields .

Customers

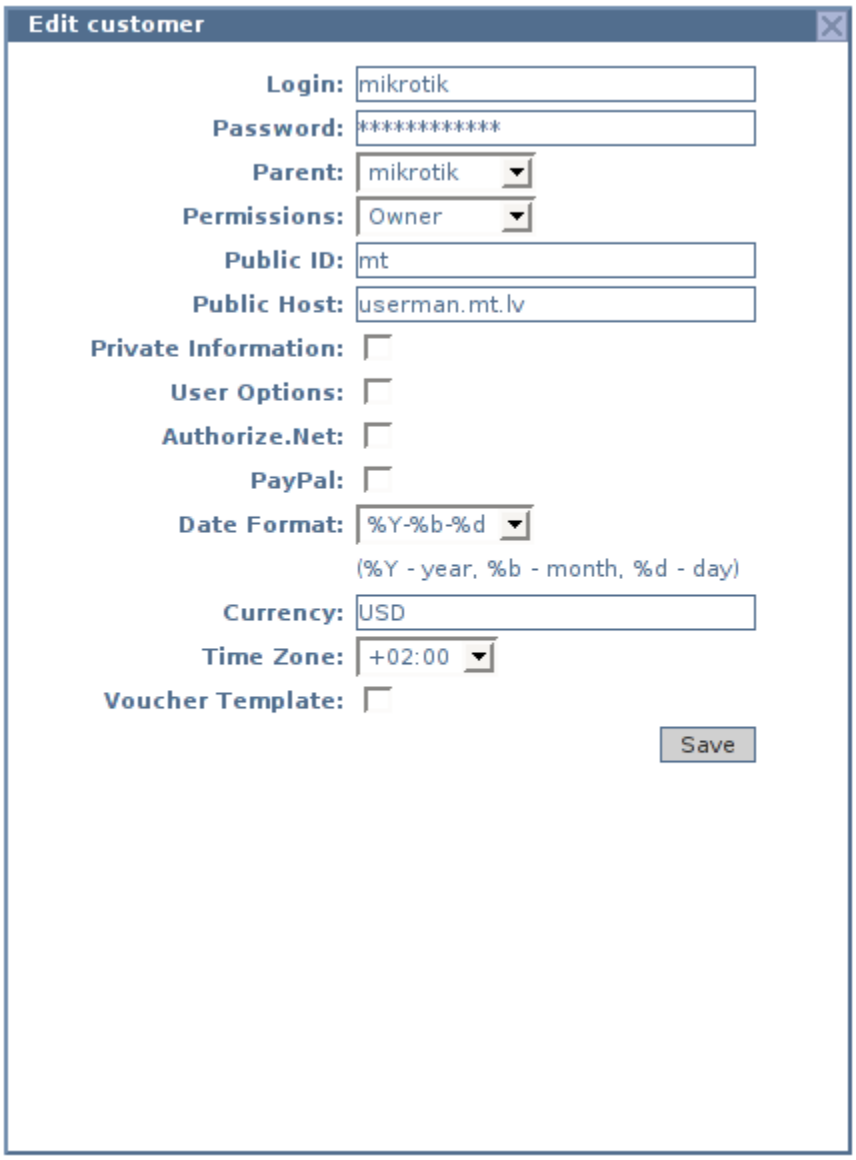
View customers

Table displaying customers:

Only part of customer's attributes are shown here. To see all details of specific customer, open customer detail form by clicking on login in the table.

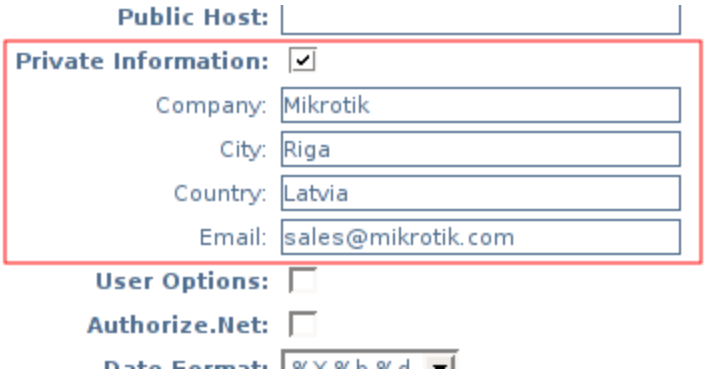
Customer detail form

Detail form with customer data:

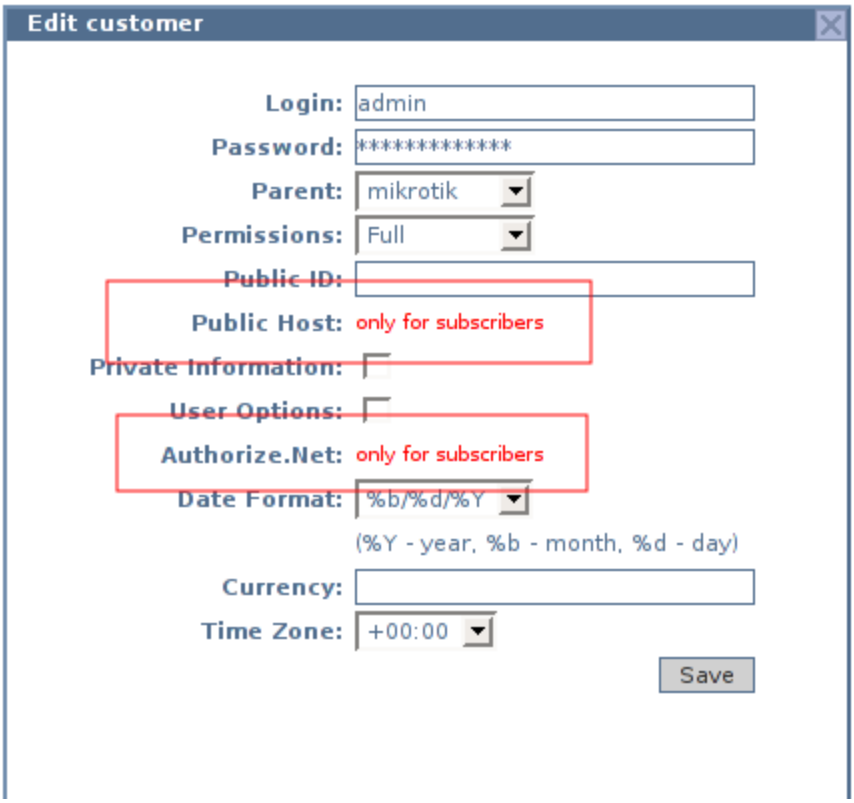


Contains all customer fields .

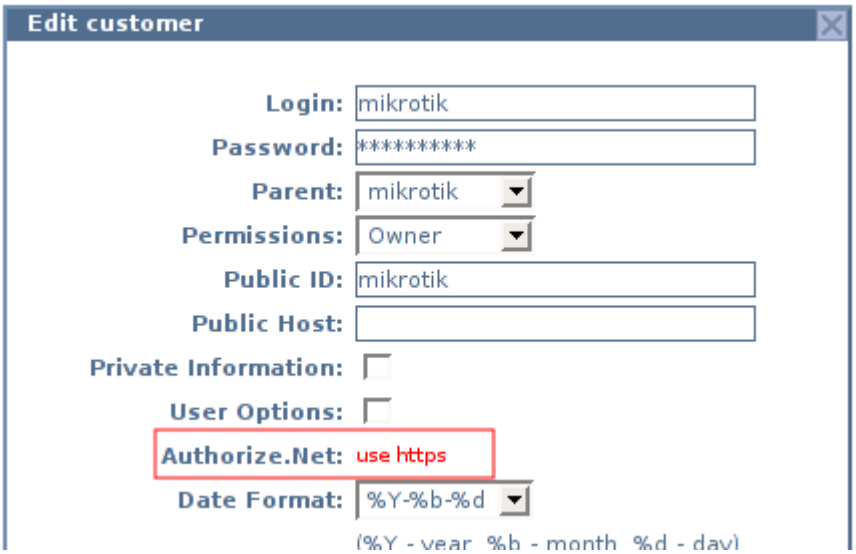
There are groups of fields (for example, private information, user options). These fields are hidden by default and are accessible by checking the box on the right:



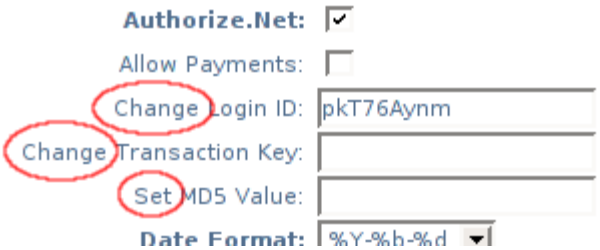
There are fields which are accessible only for subscribers: Public Host and Authorize.Net fields. These fields are not shown for customers who are not subscribers:



There are sensitive-data fields (Authorize.Net) which are visible only when using secure connection (https):



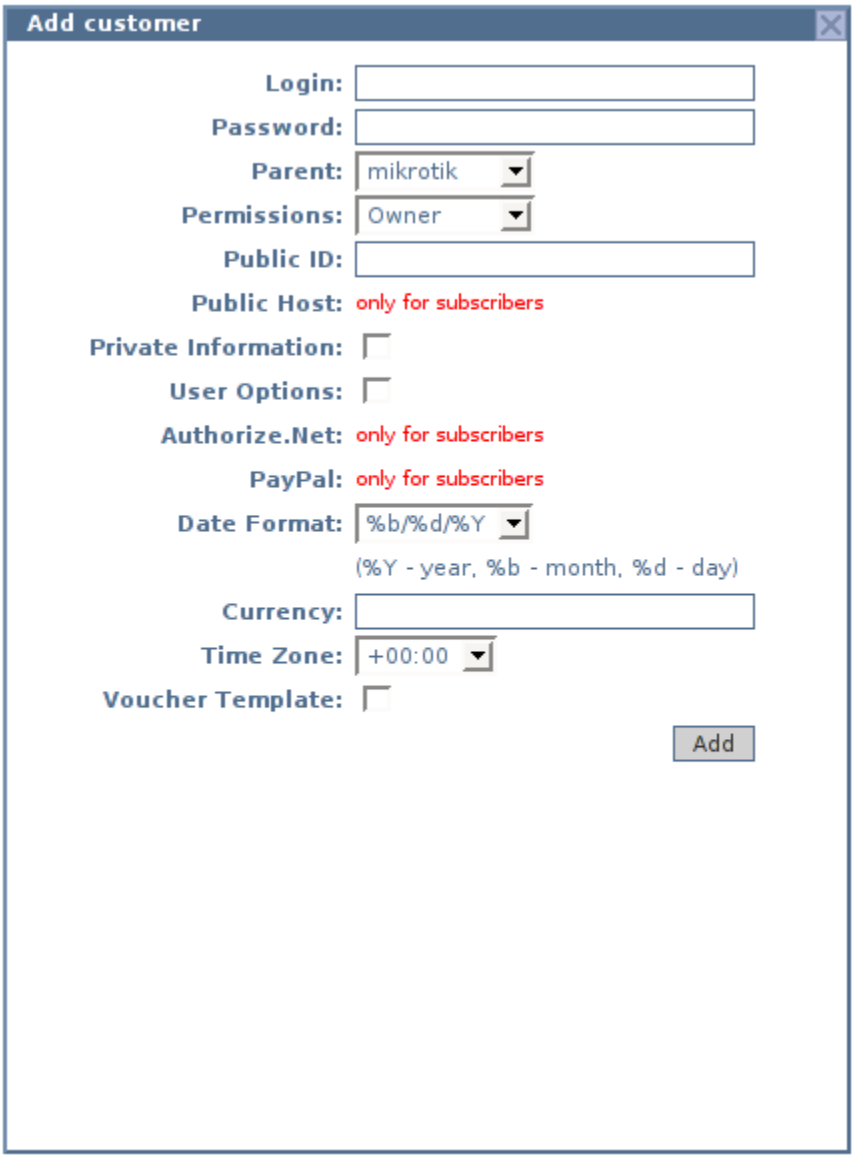
There are sensitive-data fields (Authorize.Net) whose values are not shown. Whether the field has value specified or not is visible by the title standing before it: if the title says "Set ...", this field has no value set; the title saying "Change ..." means that this field has some value:



In the example above Login ID and Transaction Key fields have values (titles are "Change ...") while MD5 Value field has no value specified (title is "Set ...").

Add customer

Detail form for filling in information about the new customer. Very similar to customer detail form. This form does not have subscriber fields since subscribers cannot be added here:



Reports

This section refers to user time and traffic reports.

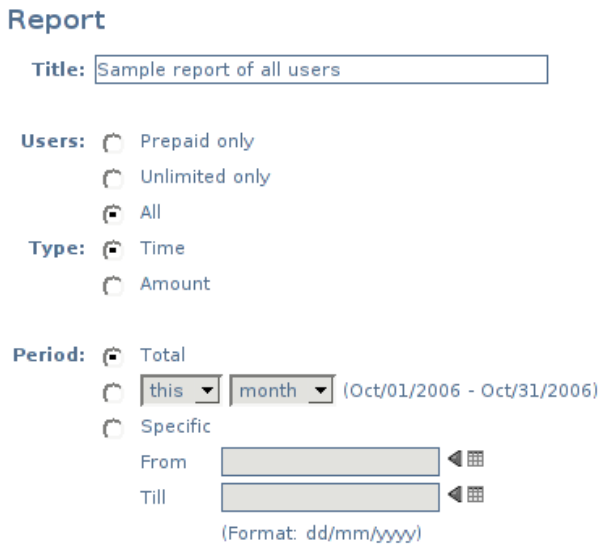
Reports generated here can be printed directly.

Configurable options:

- Users - which users to show: prepaid, unlimited or all;
- Type - time (contains prepaid time, extend time and price) or amount (contains upload and download amount) report;
- Period - total (whole history) or with specific time boundaries;

See user time and traffic reports for further detail.

Sample report:



Only this part is visible in printable form

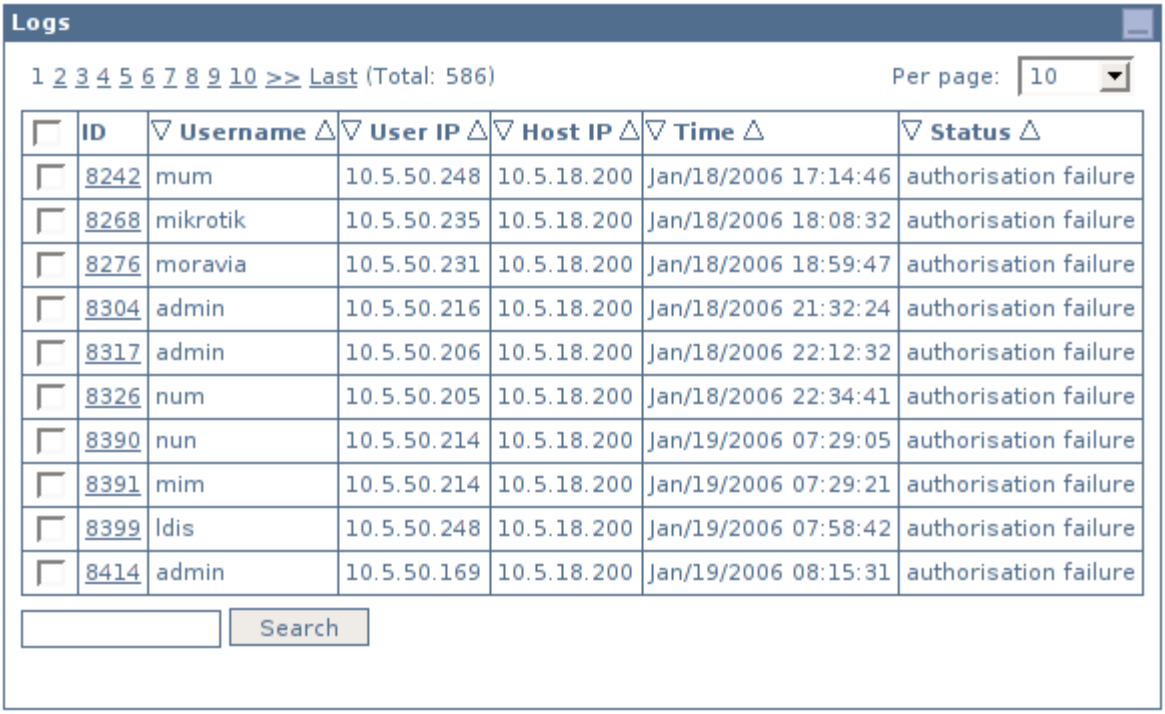
Sample report of all users

#	User	From	Till	Session Time	Prepaid Time	Extend Time	Money
1	cj2et	Jan/18/2006 17:38:38	Jan/18/2006 18:38:37	59m:28s	1h		10
				Total:	1h		10
2	pin2s	Jan/19/2006 06:33:59	Jan/19/2006 08:33:58	1h:24m:38s	2h		20
				Total:	2h		20
3	uldis	Jan/18/2006 17:12:59	Jan/19/2006 18:08:16	9h:36m:45s	unlimited		0
				Total:	0s		0
4	mum	Jan/18/2006 17:15:36	Jan/19/2006 18:48:02	1w:1d:22h:25m:8s	unlimited		0
				Total:	0s		0
				Total:	3h		30

Logs

View logs

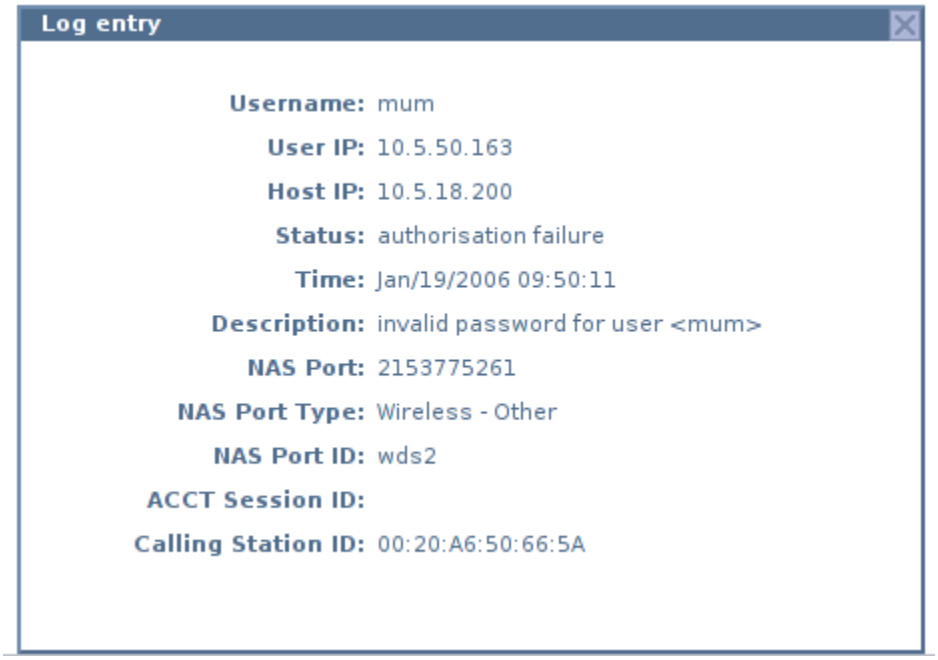
Table displaying logs:



Only part of log's attributes are shown here. To see all details of specific log, open log detail form by clicking on ID in the table.

Log detail form

Detail form with log data:



Contains all log fields .

User Manager/User page

How to find?

User page can be found at address: http://Router_IP_address/user?subs=publicID , where

- "Router_IP_address" must be replaced with IP address of your router **where the User Manager is running** (don't mix it with the HotSpot router, if User Manager and HotSpot are running on different routers);
- publicID must be replaced with public ID of the subscriber who is the owner of this user;
- If there is only one subscriber on this router the part "?subs=..." can be skipped, i.e., then the address http://Router_IP_address/user can be used.

Link to user page

Links and buttons to user page can be used in other web pages. There are several things configurable:

- router IP address;
- subscriber's public ID;
- caption on the link/button.

Textual link

To get a textual link to user page, replace this template with your own values:

```
<a href="http://%hostname%/user?subs=%subid%">%caption%</a>
```

- %hostname% - router's hostname or IP address;
- %subid% - subscriber's public ID;
- %caption% - caption of the link that will be show to user.

Example: To get a link to *userman.mt.lv* router's demo subscriber user page, use the following link:

```
<a href="http://userman.mt.lv/user?subs=demo">This is an example
link to HauteSpot User Manager demo User page</a>
```

And it looks like this: This is an example link to HauteSpot User Manager demo User page

Link button

To get a button, which leads to user page, replace this template with your own values:

```
<button
onclick="document.location='http://%hostname%/user?subs=%subid%'
">%caption%</button>
```

Example: To get a button-link to *userman.mt.lv* router's demo subscriber user page, use the following link:

```
<button
onclick="document.location='http://userman.mt.lv/user?subs=demo'
">Check</button>
```

The visual representation cannot be shown here because of the wiki security so you have to pretend how it looks like.
The same button-link is used in HotSpot page templates. By default it looks like this:

```
<!-- user manager link -->
<button
onclick="document.location='http://$(hostname)/user?subs='">stat
us</button>
```

$(hostname) here is replaced with the hostname of the HotSpot router (so the default link works only if HotSpot and User Manager are running on the same router). And "subs=" means that first subscriber will be used (works fine when there's only one subscriber on the router). Hostname and subscriber id can be replaced with desired values.

Sections

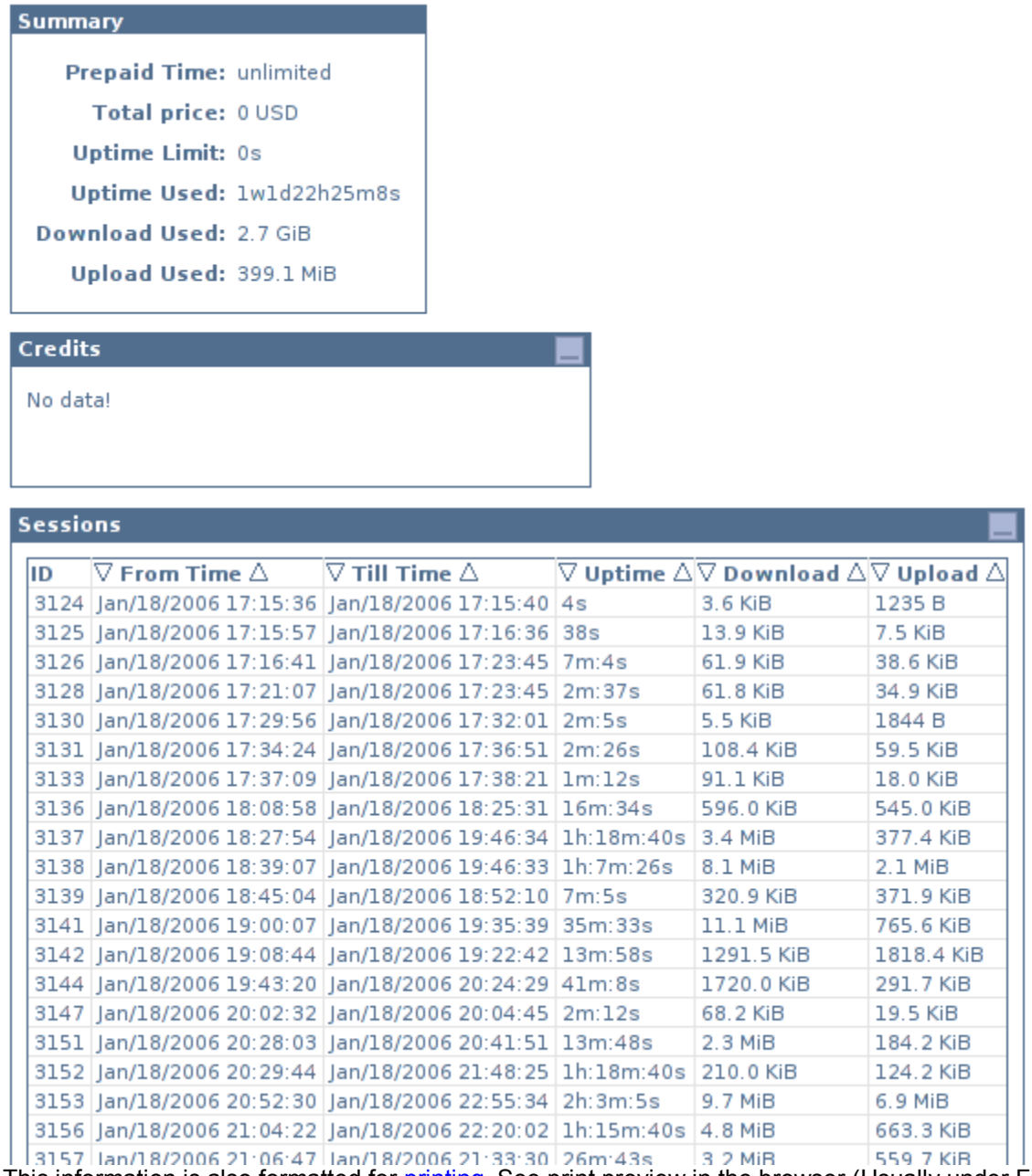
This par of a document describes sections available in user page. For navigation use the menu on the left side:

Status

Here the user can see account's status:

- Summary;
- Credits;
- Sessions.

Sample screenshot:



This information is also formatted for printing. See print preview in the browser (Usually under File > Print preview in the browser's toolbar). Credits and sessions are formed in tables. These tables can be "minimized" - the button on the upper right corner of the table. A minimized table will not be printed (see print preview).

Summary

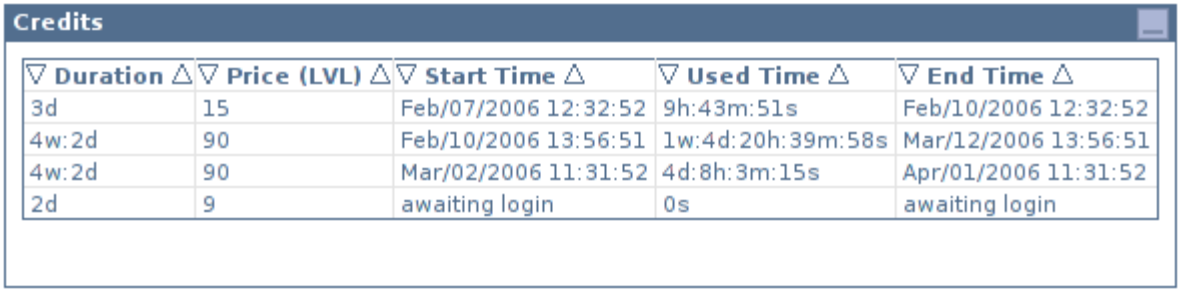
Here the user can see:

- Prepaid time - duration of all the credits bought (See: time constants). Or the word *unlimited* (See prepaid and unlimited users);
- Total price - how much all the credits cost;

- Uptime limit - the maximum allowed duration of user's sessions;
- Uptime used - current duration of user's sessions;
- Download used
- Upload used

Credits

Table with all credits this user has bought. No data for unlimited users.
Sample screenshot:

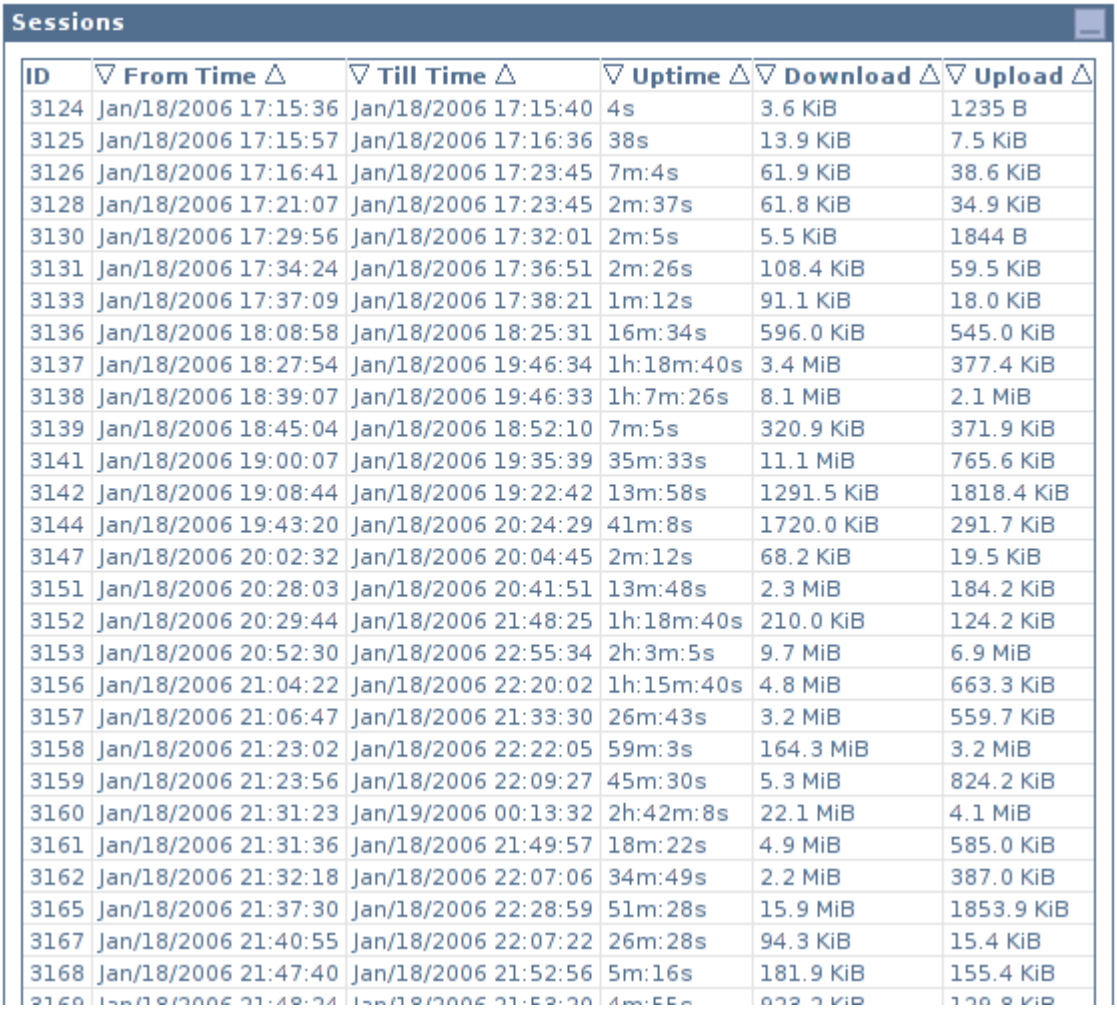


Duration	Price (LVL)	Start Time	Used Time	End Time
3d	15	Feb/07/2006 12:32:52	9h:43m:51s	Feb/10/2006 12:32:52
4w:2d	90	Feb/10/2006 13:56:51	1w:4d:20h:39m:58s	Mar/12/2006 13:56:51
4w:2d	90	Mar/02/2006 11:31:52	4d:8h:3m:15s	Apr/01/2006 11:31:52
2d	9	awaiting login	0s	awaiting login

If there are credits that are not started yet (see: credits), start-time and end-time fields contain values "awaiting login".

Sessions

Table with all user's sessions.
Sample screenshot:



Sessions

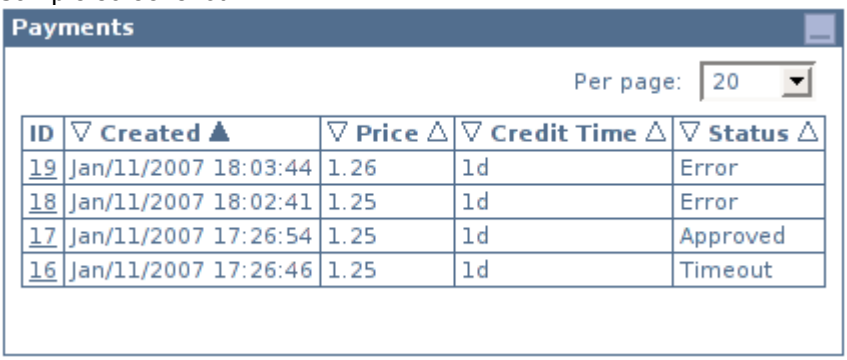
ID	From Time	Till Time	Uptime	Download	Upload
3124	Jan/18/2006 17:15:36	Jan/18/2006 17:15:40	4s	3.6 KiB	1235 B
3125	Jan/18/2006 17:15:57	Jan/18/2006 17:16:36	38s	13.9 KiB	7.5 KiB
3126	Jan/18/2006 17:16:41	Jan/18/2006 17:23:45	7m:4s	61.9 KiB	38.6 KiB
3128	Jan/18/2006 17:21:07	Jan/18/2006 17:23:45	2m:37s	61.8 KiB	34.9 KiB
3130	Jan/18/2006 17:29:56	Jan/18/2006 17:32:01	2m:5s	5.5 KiB	1844 B
3131	Jan/18/2006 17:34:24	Jan/18/2006 17:36:51	2m:26s	108.4 KiB	59.5 KiB
3133	Jan/18/2006 17:37:09	Jan/18/2006 17:38:21	1m:12s	91.1 KiB	18.0 KiB
3136	Jan/18/2006 18:08:58	Jan/18/2006 18:25:31	16m:34s	596.0 KiB	545.0 KiB
3137	Jan/18/2006 18:27:54	Jan/18/2006 19:46:34	1h:18m:40s	3.4 MiB	377.4 KiB
3138	Jan/18/2006 18:39:07	Jan/18/2006 19:46:33	1h:7m:26s	8.1 MiB	2.1 MiB
3139	Jan/18/2006 18:45:04	Jan/18/2006 18:52:10	7m:5s	320.9 KiB	371.9 KiB
3141	Jan/18/2006 19:00:07	Jan/18/2006 19:35:39	35m:33s	11.1 MiB	765.6 KiB
3142	Jan/18/2006 19:08:44	Jan/18/2006 19:22:42	13m:58s	1291.5 KiB	1818.4 KiB
3144	Jan/18/2006 19:43:20	Jan/18/2006 20:24:29	41m:8s	1720.0 KiB	291.7 KiB
3147	Jan/18/2006 20:02:32	Jan/18/2006 20:04:45	2m:12s	68.2 KiB	19.5 KiB
3151	Jan/18/2006 20:28:03	Jan/18/2006 20:41:51	13m:48s	2.3 MiB	184.2 KiB
3152	Jan/18/2006 20:29:44	Jan/18/2006 21:48:25	1h:18m:40s	210.0 KiB	124.2 KiB
3153	Jan/18/2006 20:52:30	Jan/18/2006 22:55:34	2h:3m:5s	9.7 MiB	6.9 MiB
3156	Jan/18/2006 21:04:22	Jan/18/2006 22:20:02	1h:15m:40s	4.8 MiB	663.3 KiB
3157	Jan/18/2006 21:06:47	Jan/18/2006 21:33:30	26m:43s	3.2 MiB	559.7 KiB
3158	Jan/18/2006 21:23:02	Jan/18/2006 22:22:05	59m:3s	164.3 MiB	3.2 MiB
3159	Jan/18/2006 21:23:56	Jan/18/2006 22:09:27	45m:30s	5.3 MiB	824.2 KiB
3160	Jan/18/2006 21:31:23	Jan/19/2006 00:13:32	2h:42m:8s	22.1 MiB	4.1 MiB
3161	Jan/18/2006 21:31:36	Jan/18/2006 21:49:57	18m:22s	4.9 MiB	585.0 KiB
3162	Jan/18/2006 21:32:18	Jan/18/2006 22:07:06	34m:49s	2.2 MiB	387.0 KiB
3165	Jan/18/2006 21:37:30	Jan/18/2006 22:28:59	51m:28s	15.9 MiB	1853.9 KiB
3167	Jan/18/2006 21:40:55	Jan/18/2006 22:07:22	26m:28s	94.3 KiB	15.4 KiB
3168	Jan/18/2006 21:47:40	Jan/18/2006 21:52:56	5m:16s	181.9 KiB	155.4 KiB

Payments

Here the user can view payment history and buy a new credit. This section is only available if the subscriber has allowed any payments.

View payments

Table with all user payments.
Sample screenshot:



Payments

Per page: 20

ID	Created	Price	Credit Time	Status
19	Jan/11/2007 18:03:44	1.26	1d	Error
18	Jan/11/2007 18:02:41	1.25	1d	Error
17	Jan/11/2007 17:26:54	1.25	1d	Approved
16	Jan/11/2007 17:26:46	1.25	1d	Timeout

To see all details of specific payment, open payment detail form by clicking on ID in the table.

Payment detail form

Detail form with payment data:

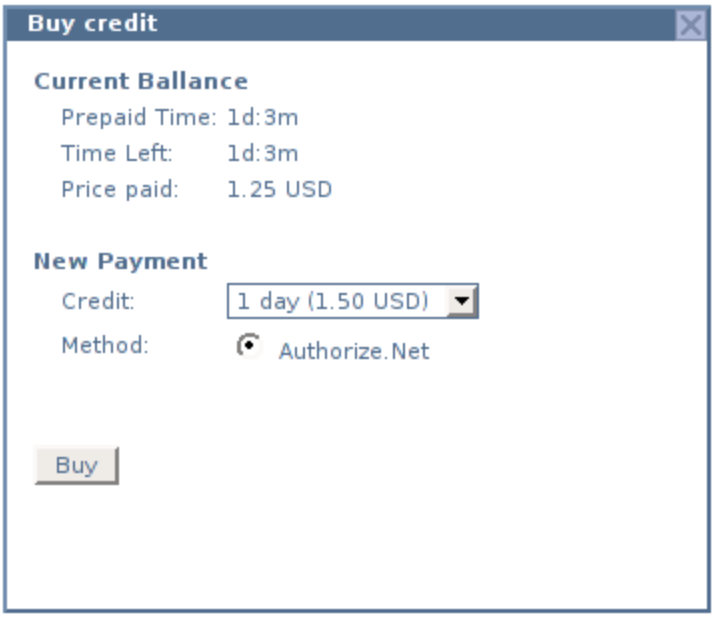
Contains all payment fields.

Buy credit

A new credit can be bought here using payment methods which are allowed by the subscriber. There are a number of restrictions for this sub-section to be accessible:

- Secure connection (https) must be used to access the site. Otherwise a notification with a link to secure page will be shown;
- At least one payment method must be allowed by the subscriber;
- Subscriber must have configured all required payment attributes;

Sample screenshot:



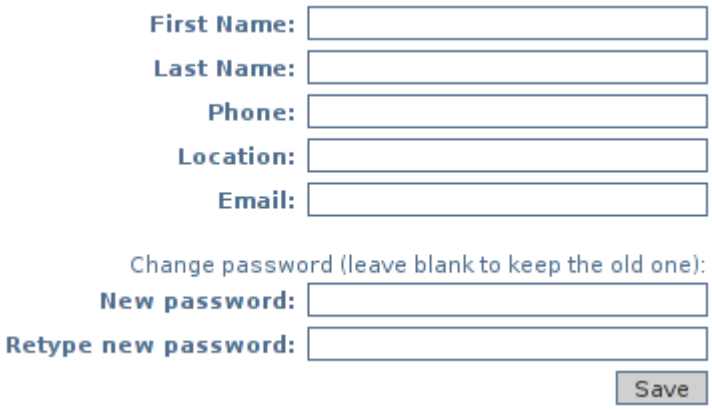
Here user can see his/her current balance and choose a credit to buy. After click on the "Buy" button user will be redirected to payment gateway where he/she will have to enter required data to process payment.
Important - payment data (such as credit card number and expiry date) is sent directly from user's computer to payment gateway and is not captured by User Manager. User Manager processes only response about the payment result from the payment gateway. This response does not contain any sensitive user's data.
When the payment is successful, the selected credit is added to user's account.

Settings

In this section user can configure his/her parameters:

- Private information (informational, not used by User Manager):
  - First name;
  - Last name;
  - Phone;
  - Location.
- Email - used to send emails to user. Must be unique.

If values provided in "New password" and "Retype new password" fields, the password will be changed.
Sample screenshot:



User Manager/User sign up

Setup

User sign-up can be enabled per customer. I.e., some customers can allow it while others don't. Sign-up is disabled by default. To enable it several requirements must be met:

- ***Note:*** *All the attributes mentioned above can be configured in customer section of the customer web-page*;

- Customer, who wants to allow sign-up, **must** have **public ID**. Since Only subscribers have permissions to edit customers, this public ID must be assigned by the subscriber. In other words - subscriber must configure public IDs for its customers.

- Subscriber **must** have at least one credit with *full price* specified;
- In the case when users access sign-up page from a local address which is not accessible from outside (global Internet) subscriber **must** have **public host** address configured. This address is needed by PayPal, payment response will be sent to this it;
- The customer **has to** enable sign-up by checking the **"Signup allowed"** box in *Signup options* section;
- The subscriber **must** have at least one **payment method** enabled and configured;
- The customer **should** have **email address** specified. Email will be send to users who sign up (if the user specifies his/her email address) using this as the *from-address*;
- **SMTP-server should** be specified. It can be done via console, under tool email, command "set server=xxx.xxx.xxx.xxx". This SMTP server will be used to send email reminding user's account data. Users can however log on to the HotSpot after a successful payment without receiving this email;
- **Signup email subject and body can** be personalized. There are defaults defined, but one can customize them. However there are constant strings (will be replaced by actual values) that must be present within the message body. See *sign-up email body* field definition.

Sign-up steps

User sign-up can divided in following steps:

- Subscriber configures required parameters (described above);
- User creates an account:
  - User opens sign-up page URL in the browser;
  - User fills in the sign-up form;
  - User chooses credit;
  - User chooses payment method;
  - An inactive account is created for the user;
- User activates the account (executes payment):
  - User is redirected to Payment Gateway;
  - The payment is being processed;
  - Payment gateway sends response (was the payment successful or not) to User Manager router;
  - The account gets activated (if the payment was successful);
- User can start using services. Status check and setting change can be done in the user web-page.

May seem a little confusing, but all these steps are simple and can be done in several minutes.

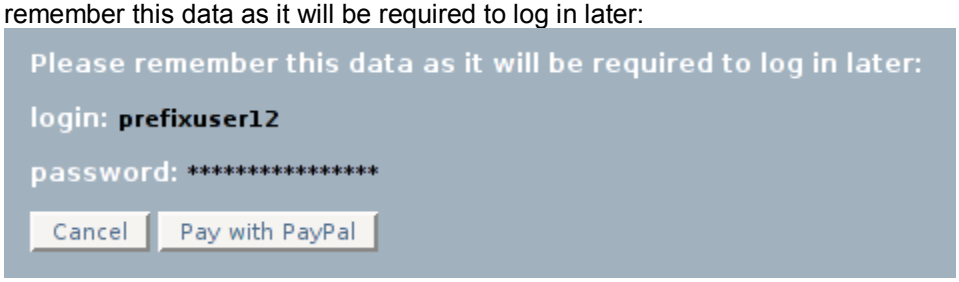
Creating account

User opens http://routerIP/user?signup=publicID, where *routerIP* must be replaced with the IP address of the User Manager router and *publicID* must be replaced with subscribers public id. Sign-up form will be shown:

Input fields:

- email. Email address for user account. must be unique per subscriber. Account data will be sent to this address if one specified;
- login. Desired username. If user prefix is defined, it is shown at the left and cannot be changed. So the prefix is already predefined (may be empty), the remaining part of username can be chosen. IT must be at least 3 characters long. Example: if the prefix is "cu" (shown on the left) and "test" is entered as the remaining part, the username will be "cutest";
- password. Self explanatory;
- confirm password. Password once again to reduce possibility to mistype it;
- time. The initial credit for the user account;
- pay with. Payment method selector.

After the "sign up" button is pressed, authorization data is show to the user. He/She must remember this data as it will be required to log in later:



If the "Cancel" button is pressed, user is returned to sign-up form.
If the "Pay with ..." button is pressed, an inactive account is being created and the user is redirected to payment gateway.

Activating account

On a successful payment, the account is activated and the user is returned to User Manager/User page where he/she can check the status of the account.

If the email address was specified in sign-up form, an email with authorization information is sent to it. The text is customizable in customer web-page. By default it looks like this:

```
Your authorization data:
login: userLogin
password: userPassword

To check your status and buy extented time go to address
http://userman.mt.lv/user?subs=demo.
```

here:

- *userLogin* is the username (login);
- *userPassword* is the password.
- *http://userman.mt.lv/* is the hostname of the User Manager router;

Login

After successful account activation user is able to start using services (Hotspot). Status and settings are available in user web-page.

User Manager/User payments

Supported payment methods

Authorize.Net (since version 2.9.40 or 3.0beta5) and PayPal (since version 2.9.41 or 3.0beta6) payments are supported.

Authorize.Net

Authorize.Net requirements

To allow Authorize.Net payments for users the following requirements must be met:

- User Manager v3.0 (or v2.9.x, >= 2.9.40) package installed on the router. See: Getting started;
- User Manager subscriber created (See: Getting started);
- Subscriber must have merchant account in Authorize.Net gateway;
- Web server on the router must be configured to support secure SSL connections (See HTTPS connection enabling);
- HotSpot router should contain entries in 'walled-garden to User Manager router and Authorize.net webpage,

```
/ ip hotspot walled-garden ip add dst-address=x.x.x.x
action=accept
```

where x.x.x.x is address of User-Manager server,

```
/ ip hotspot walled-garden add dst-
host=:^secure\\.authorize\\.net dst-port=443 action=allow
```

These entry is used to allow access to Authorize.net

Authorize.Net setup

Authorize.Net merchant account configuration

Relay URL

Relay URL list must either be empty or contain URL to the User Manager router. For example, if you are using userman.mt.lv as User Manager router, then Relay URL list must contain URL https://userman.mt.lv/ (works with and without trailing slash). Relay URL list can be configured in Authorize.Net merchant gateway under Account > Settings > Response/Receipt URLs

API Login ID

API Login ID is shown in Authorize.Net merchant gateway under Account > Settings > API Login ID and Transaction Key.

Transaction Key

Transaction Key can be obtained in Authorize.Net merchant gateway under Account > Settings > Obtain Transaction Key.

MD5-Hash value

MD5-Hash value can be set in Authorize.Net merchant gateway under Account > Settings > MD5-Hash.
WARNING!: Standard MD5 hash values are 32 characters long, however, the Authorize.net MD5-Hash input fields only allow 20 characters. Best chance of success if you paste your md5sum into the Authorize.net input field, then copy it back out to paste into User Manager configuration. By re-copying from the Authorize.net input field, you are selecting only the 20 characters that the field length allows.

Payment Form

Payment Form configuration can be found in Authorize.Net merchant gateway under Account > Settings > Payment Form. The look of this form is customizable here. While the only required fields for processing transaction are credit card number and expiration date, another fields are allowed to be shown in the form. Form customization is up to merchant.

Authorize.Net subscriber configuration

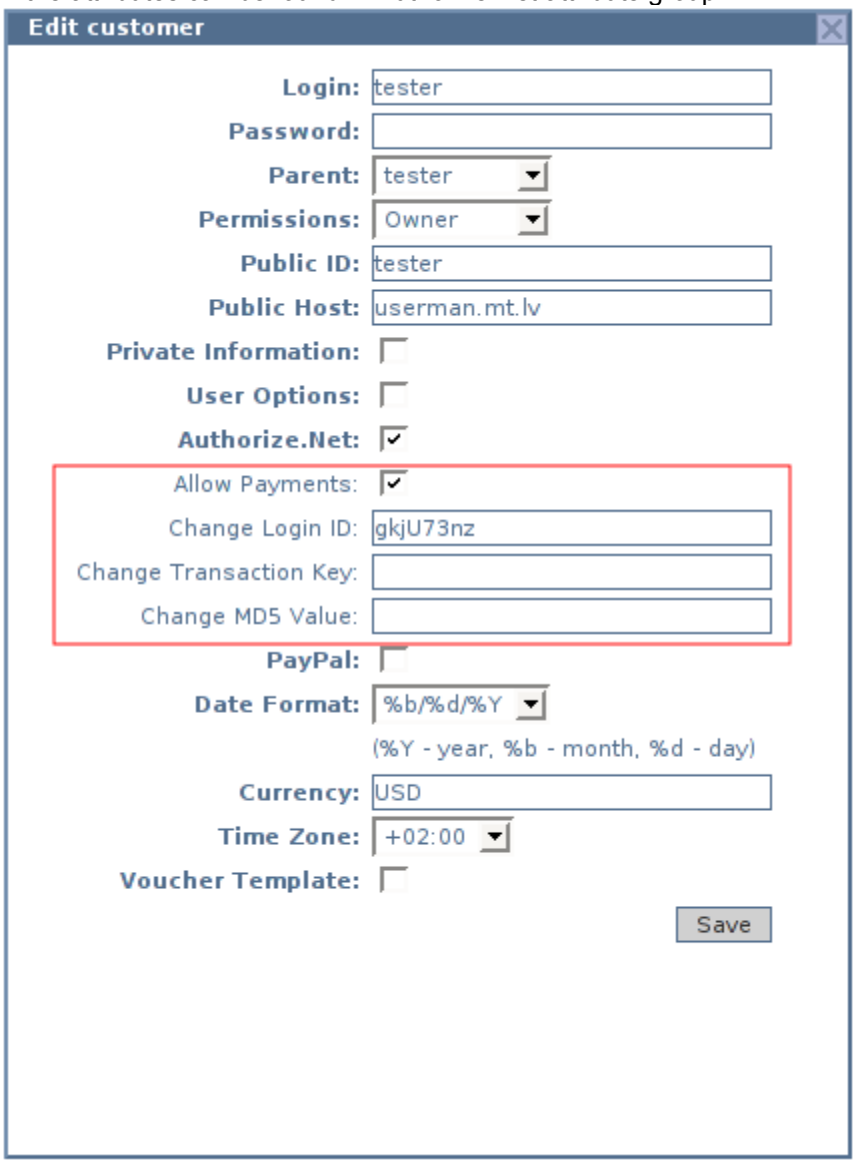
Subscriber attribute values can be edited using customer detail form in customer page.

Subscriber Authorize.Net attributes

Subscribers have a set of specific Authorize.Net attributes which must be configured properly to allow Authorize.Net payments:

- Only subscribers have Authorize.Net attributes, other customers don't;
- Attribute values can be changed only in customer web page, not in console. There is only possibility to change values, not to see them. As these attributes contain sensitive data, their values are encrypted on the router;
- Customer web page must be opened using secure SSL connection (https) to change attribute values;

All the attributes can be found in Authorize.Net attribute group:



1. "Allow Payments" must be checked to allow this payment method;
2. Login ID, Transaction Key and MD5 Value must have same values as set in Authorize.Net merchant gateway.
3. Title is optional. It specifies the text shown to users as the name of this payment method. Default title is "Authorize.Net", but it can be changed to something more used to users, for example "Credit Card". The value of this field **does not** affect the payment process it is only user interface element.

Other subscriber requirements

- Subscriber must have at least one credit with price other than zero. Credit price will be used as transaction amount for the payment;

- Correct currency must be specified for subscriber. If USD is accepted by Authorize.Net merchant, currency attribute can be left unchanged for subscriber:

- If users access User Manager page through a local IP address, public host attribute must be specified. It must contain a public address of User Manager router which is acceptable as Relay URL for Authorize.Net gateway (See: Authorize.Net Merchant account configuration). Domain name or IP address can be used. Only the address must be specified, not URL (for example, userman.mt.lv, not https://userman.mt.lv/ and not https://userman.mt.lv/userman):

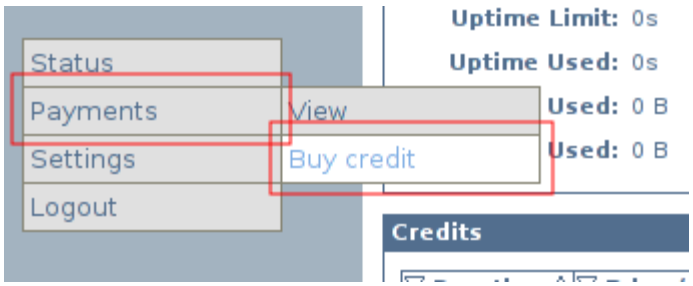
Authorize.Net usage

- User can buy credits in User Manager page. First he/she has to log on the page. See: User page.

- Secure connection must be used for web page, so user has to use https://router_IP/user instead of http://router_IP/user (https instead of http).

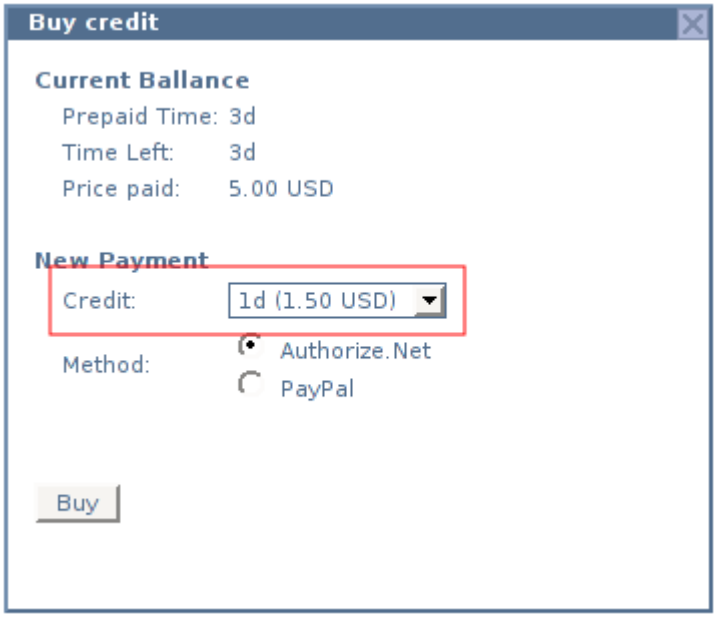
- Payment section is available on main menu only if subscriber has allowed any payment method.

- To buy credit user chooses "Buy credit" from "Payments" section:

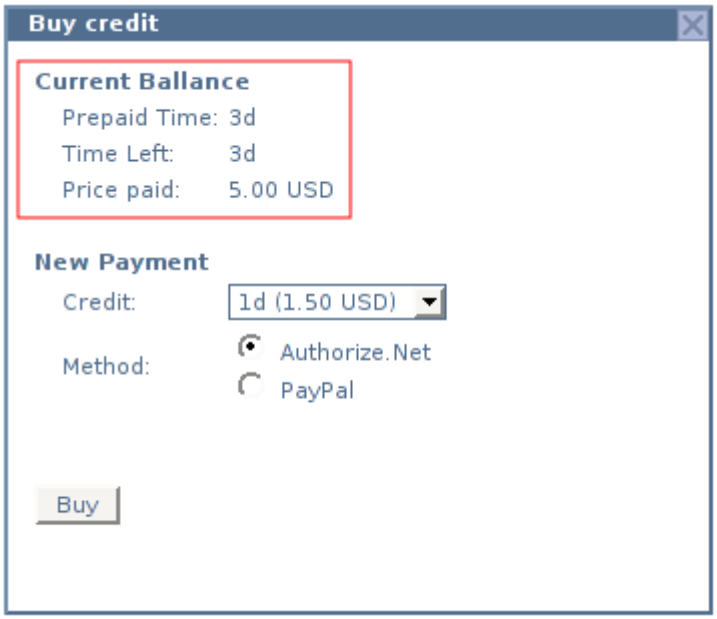


- If https connection is not used for web session, a message with error and link to https site will be opened:

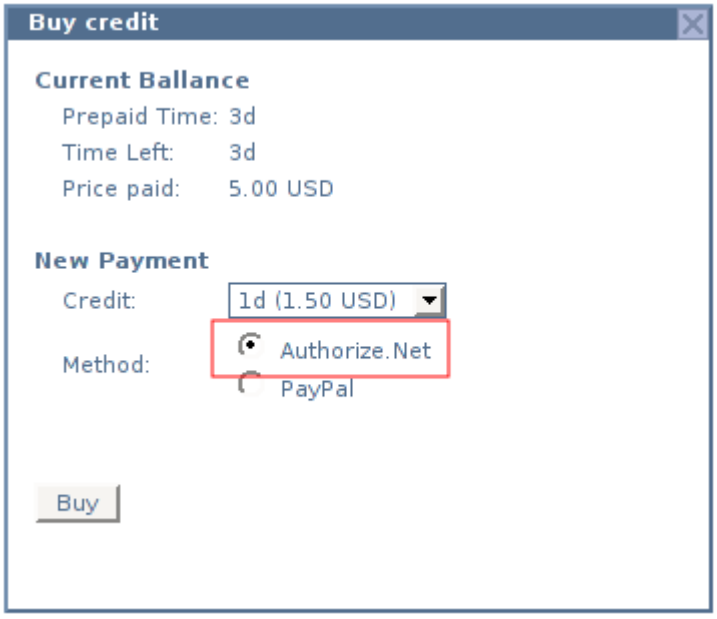
- In this form user chooses credit he/she wishes to buy;



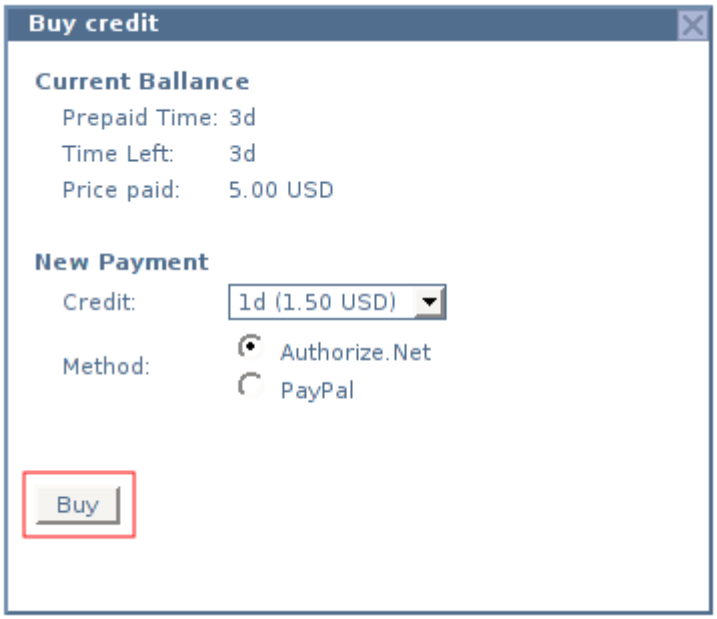
- Current balance is also shown:



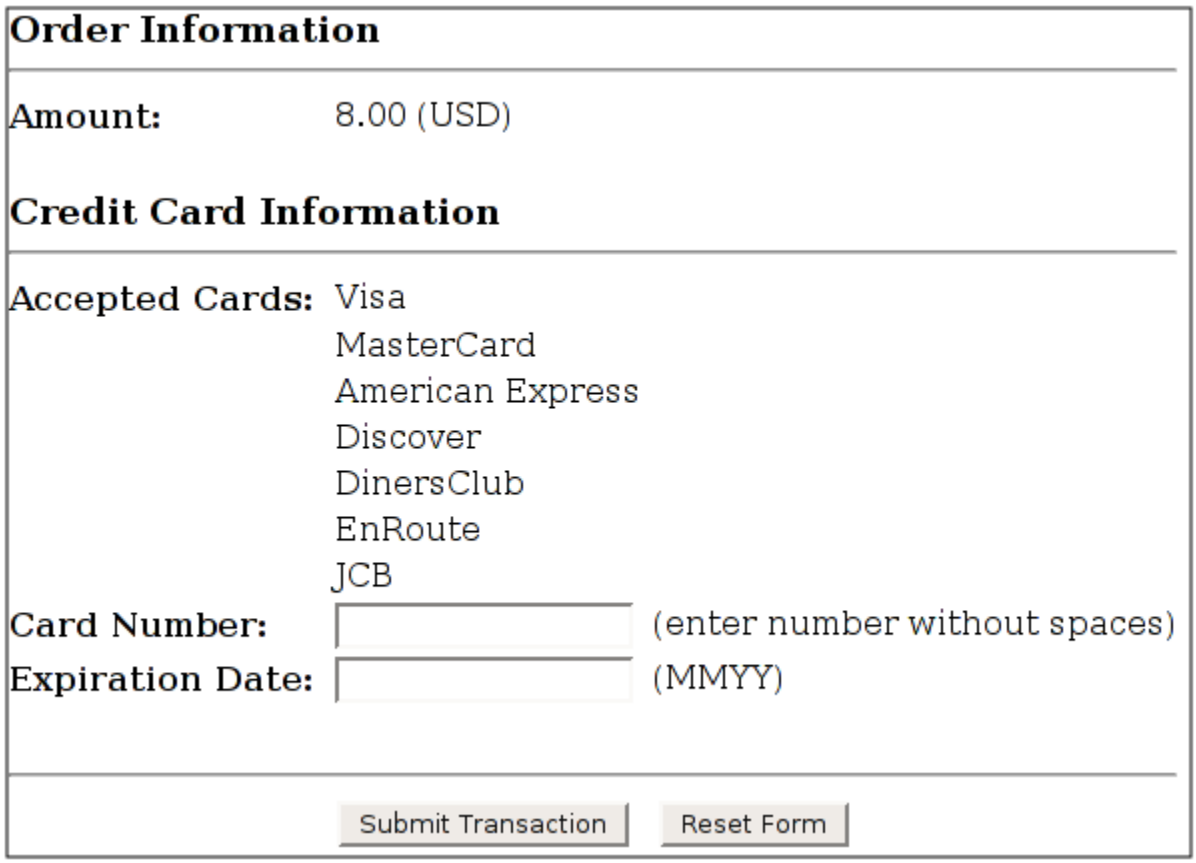
- User chooses Authorize.Net as payment method:



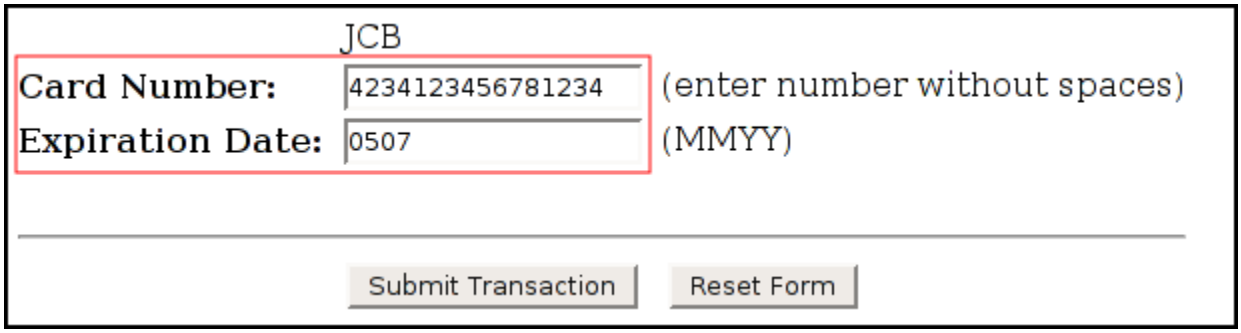
- When the credit is chosen, "Buy" button must be pressed to start payment transaction:



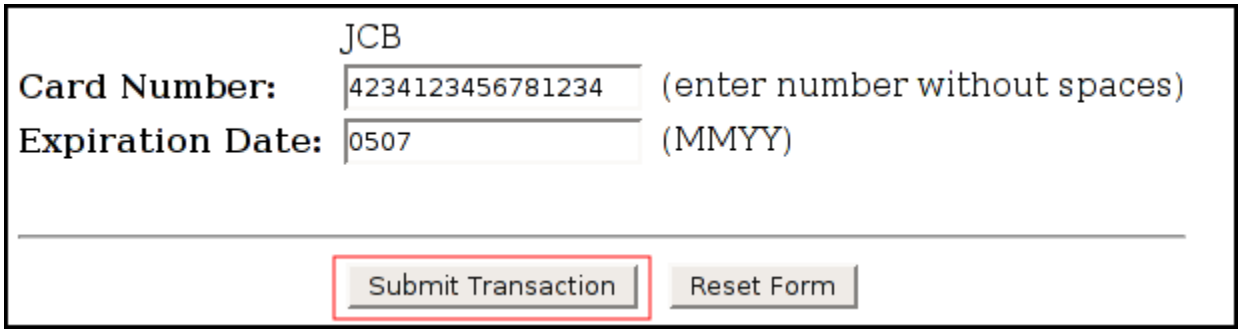
- User is redirected to Authorize.Net gateway payment form, which should look similar to following:



- The actual look of this form can be configured in Authorize.Net merchant gateway
- User fills in credit card number and expiry date. Other fields are optional:



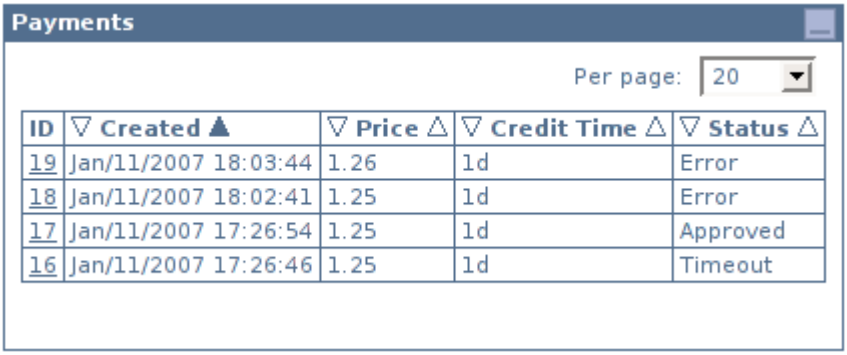
- User submits the form::



- The data is transmitted directly to Authorize.Net gateway via secure connection. Neither credit card number nor expiry date is submitted to User Manager router.
- Authorize.Net gateway processes the data and sends response to specified User Manager router. This response contains only data required to identify payment in User Manager and detect result status of transaction - was it successful or not. It does not contain any information about the user - credit card number, expiry date or other sensitive data.
- User Manager processes the response and updates payment record status;
- If the transaction was successful requested credit is added to user's account;
- A message describing payment result is shown to user:

- Click on the button redirects the user back to User Manager page:

- User is returned to payment section displaying table with payment history:


Payments
Per page: 20
ID ▽ Created ▲ ▽ Price △ ▽ Credit Time △ ▽ Status △
19 Jan/11/2007 18:03:44 1.26 1d Error
18 Jan/11/2007 18:02:41 1.25 1d Error
17 Jan/11/2007 17:26:54 1.25 1d Approved
16 Jan/11/2007 17:26:46 1.25 1d Timeout

PayPal

PayPal requirements

To allow PayPal payments for users the following requirements must be met:

- User Manager v3.0 (>= 3.0beta6) or v2.9.x (>= 2.9.41) package installed on the router. See: Getting started;
- User Manager subscriber created (See: Getting started);
- Subscriber must have merchant PayPal account;
- Web server on the router must be configured to support secure SSL connections (See HTTPS connection enabling);
- HotSpot router should contain entries in 'walled-garden to User Manager router and Paypal webpage,

```
/ ip hotspot walled-garden ip add dst-address=x.x.x.x
action=accept
```

where x.x.x.x is address of User-Manager server;

- **version v2.9**

```
/ ip hotspot walled-garden add dst-host=:^www\\.paypal\\.com\$
dst-port=443 action=allow
/ ip hotspot walled-garden add dst-
host=:^content\\.paypalobjects\\.com\$ dst-port=443 action=allow
```

```
/ ip hotspot walled-garden add dst-host=*.akamaiedge.net
action=allow
```

- **version v3**

```
/ ip hotspot walled-garden add dst-host=":^www\\.paypal\\.com\$"
dst-port=443 action=allow
/ ip hotspot walled-garden add dst-
host=":^content\\.paypalobjects\\.com\$" dst-port=443
action=allow
/ ip hotspot walled-garden add dst-host=*.akamaiedge.net
action=allow
```

These two entries are used to allow access to Paypal system.

PayPal setup

PayPal merchant account configuration

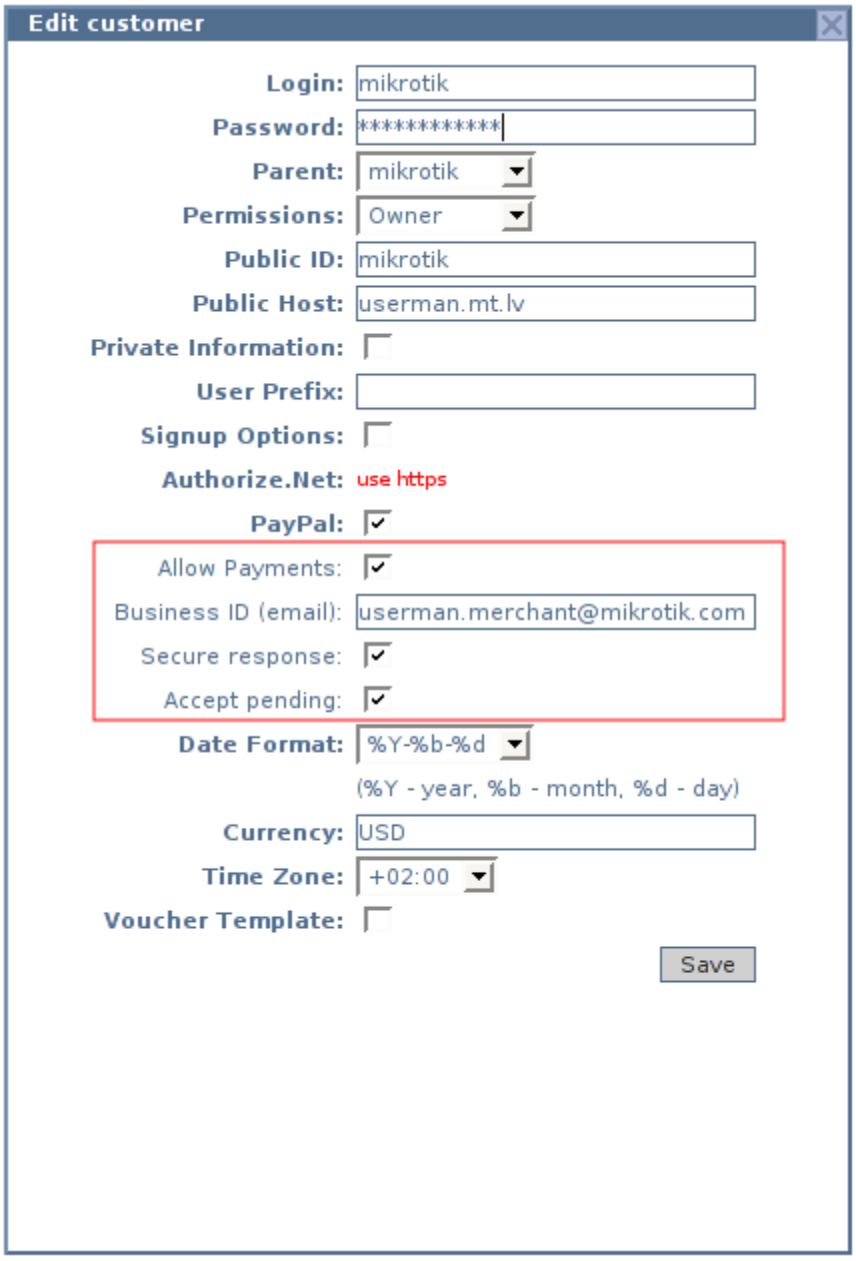
Basically there is no specific PayPal account configuration that must be done. The only requirement is to have PayPal account which is allowed to receive money.
Warning! User Manager accepts payment as successful only when it receives status "Completed" from PayPal gateway. If the status is "Pending" and some manual operations must be done by merchant (or the merchant has not verified the account) to accept payment, the credit will be transfered to User Manager user account only when the payment will be accepted.
Note: Since version 2.9.45 and 3.0beta11 it is possible to also accept payments with "Pending" status, except for those with pending reason "unilateral".

PayPal subscriber configuration

Subscriber attribute values can be edited using customer detail form in customer page.

Subscriber PayPal attributes

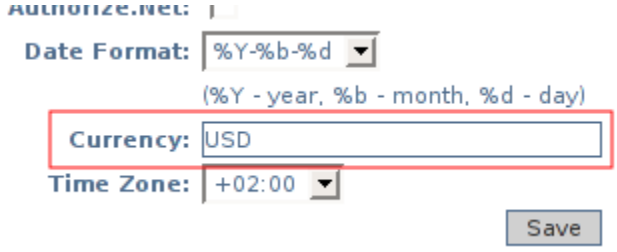
The only PayPal attribute subscribers have is business login. It is the login (usually an email address) merchants use to log on their account. Only subscribers have this business login, other customers don't;
Since versions 2.9.45 and 3.0beta11 there are also options that refer to PayPal payment processing: "Secure Response" and "Accept Pending".
Field "Return URL" added in version 3.11.
All the attributes can be found in PayPal attribute group:



1. "Allow Payments" must be checked to allow this payment method;
2. Login (email) must be the PayPal merchant account login.
3. Secure response. When checked, PayPal will send response via HTTPS. Otherwise response will be send via HTTP;
4. Accept pending. When checked, User Manager will also add credit to user if the payment status is "Pending", except for payments with pending reason "unilateral".

Other subscriber requirements

- Subscriber must have at least one credit with price other than zero. Credit price will be used as transaction amount for the payment;
- Correct currency must be specified for subscriber. If USD is accepted by PayPal merchant, currency attribute can be left unchanged for subscriber:



- If users access User Manager page through a local IP address, public host attribute must be specified. It must contain a public address of User Manager router which is acceptable as response URL for PayPal gateway (PayPal will send payment result to this address). Domain name or IP address can be used. Only the address must be specified, not complete URL (for example, userman.mt.lv, not https://userman.mt.lv/ and not https://userman.mt.lv/userman):

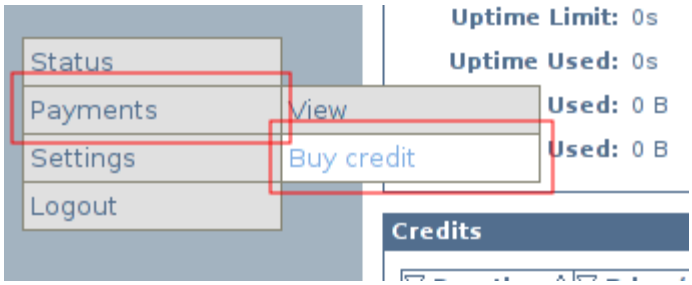
PayPal usage

- User can buy credits in User Manager page. First he/she has to log on the page. See: User page.

- Secure connection must be used for web page, so user has to use https://router_IP/user instead of http://router_IP/user (https instead of http).

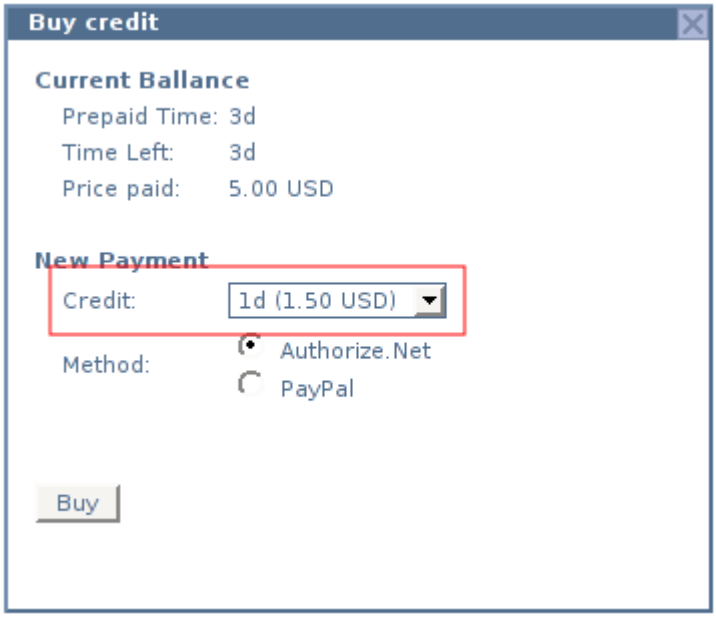
- Payment section is available on main menu only if subscriber has allowed any payment method.

- To buy credit user chooses "Buy credit" from "Payments" section:

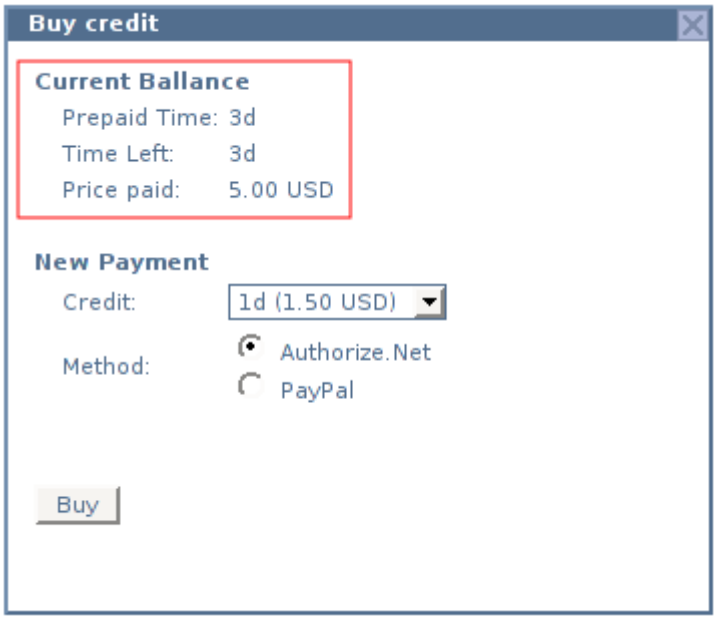


- If https connection is not used for web session, a message with error and link to https site will be opened:

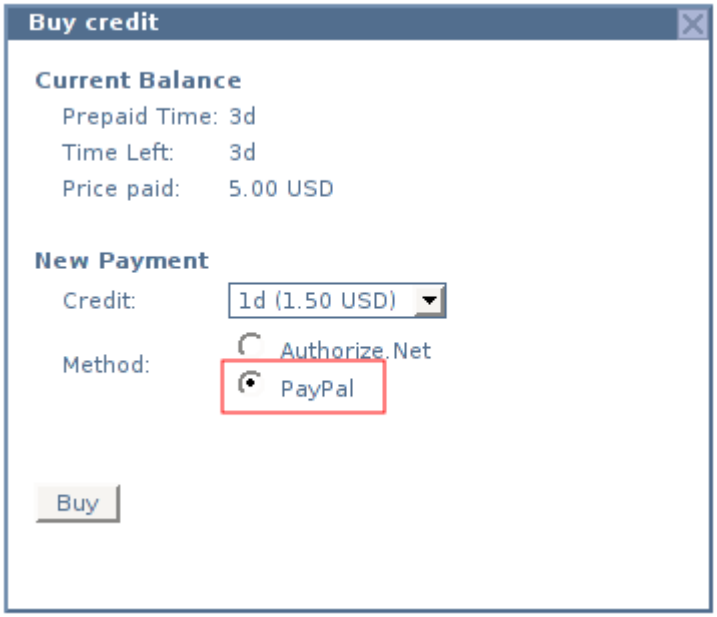
- In this form user chooses credit he/she wishes to buy;



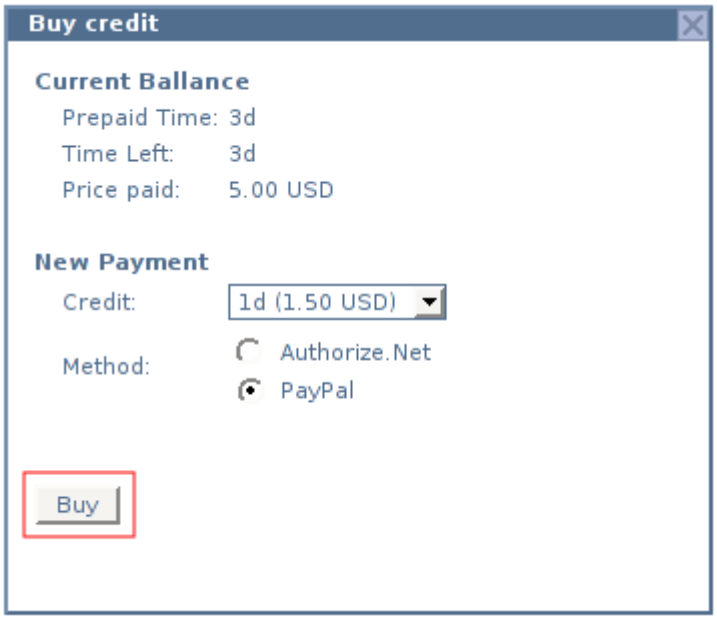
- Current balance is also shown:



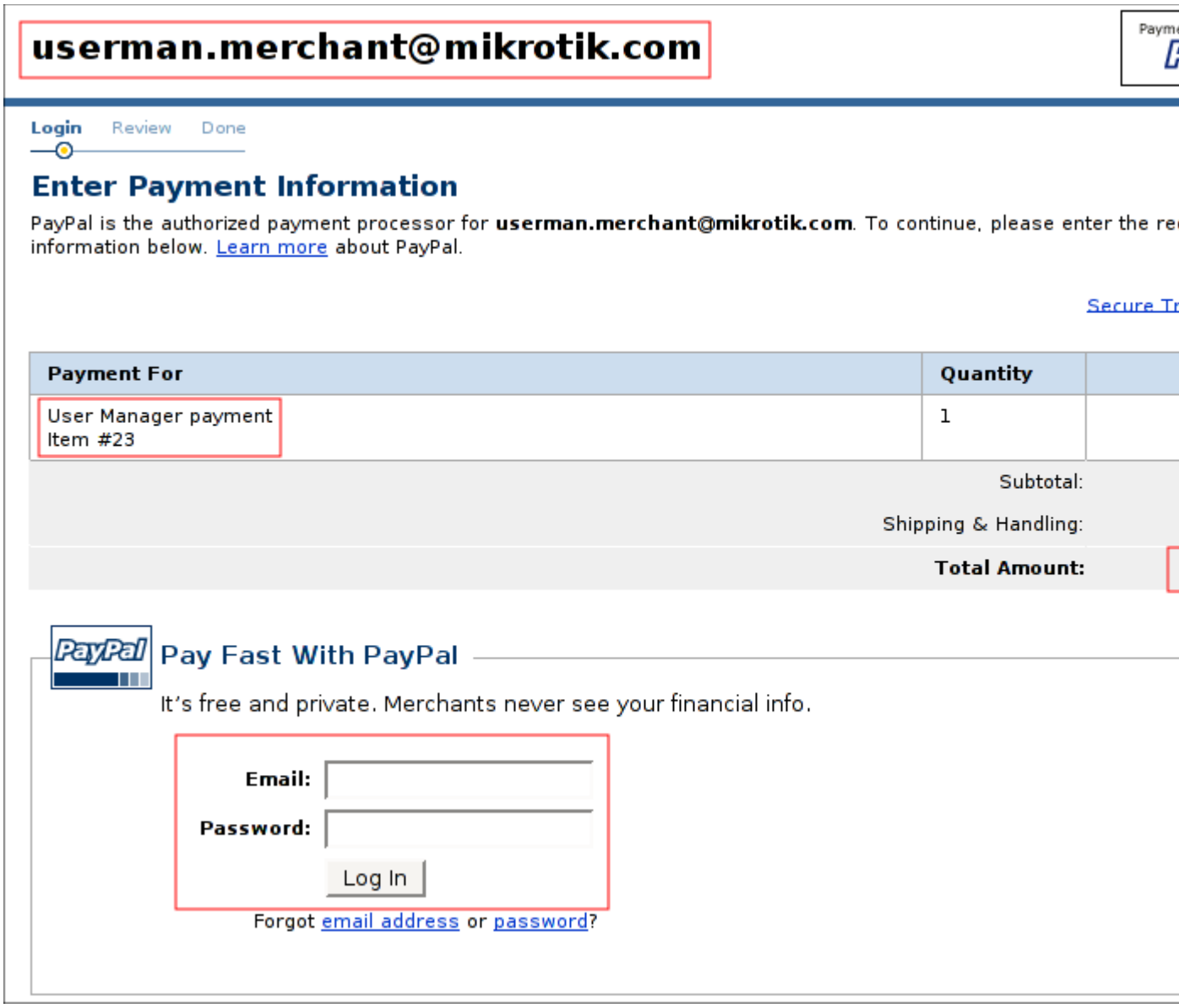
- User chooses PayPal as payment method:



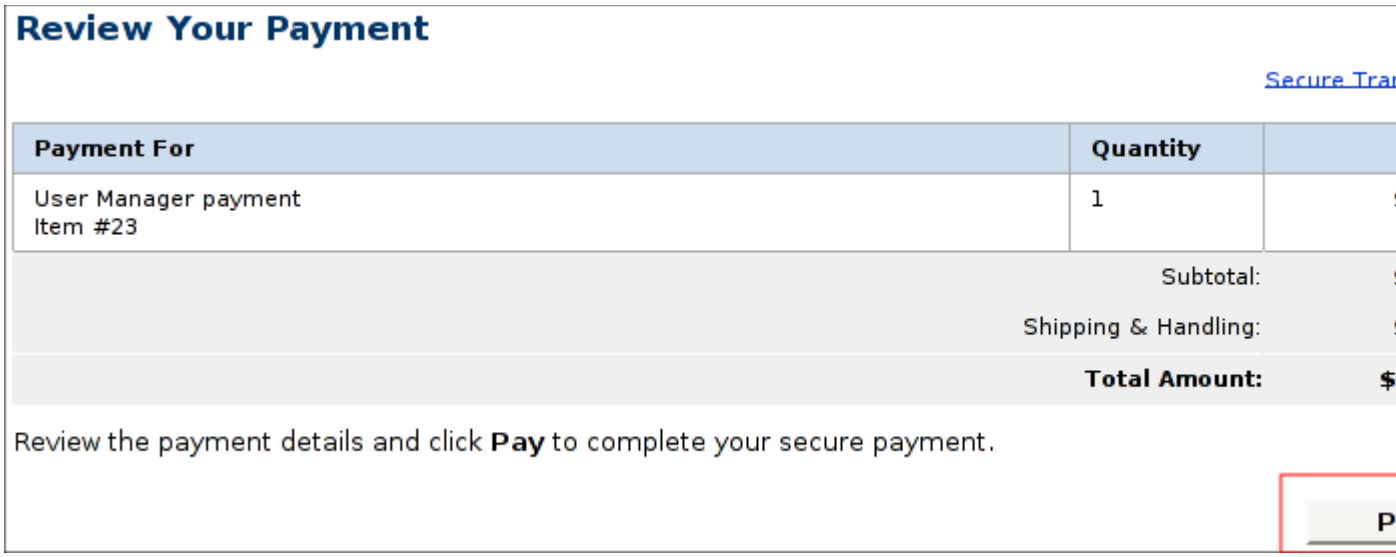
- When the credit is chosen, "Buy" button must be pressed to start payment transaction:



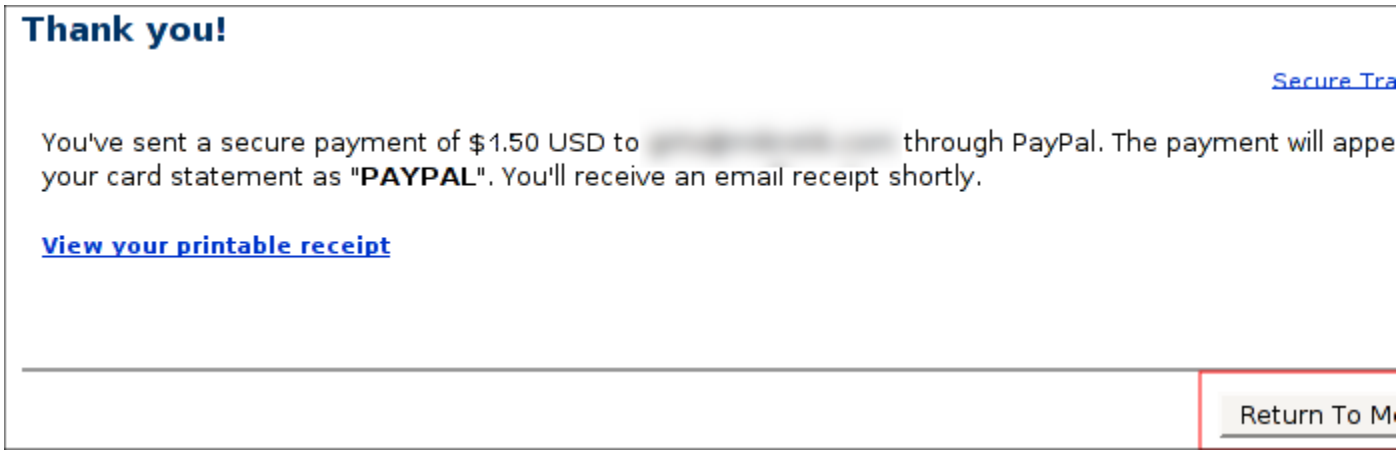
- User is redirected to PayPal gateway payment form, which should look similar to following (PayPal web site can change, these screen shots may differ from actual page):



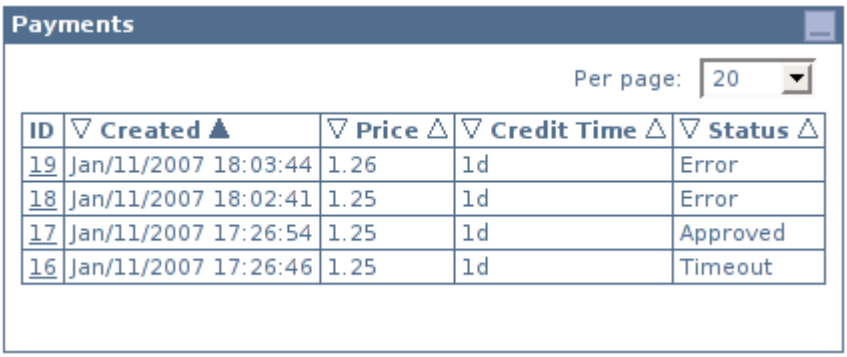
- User logs on to the account. Payment is now displayed with the Pay button:



- When user presses Pay button, PayPal starts to process data. On successful payment result page is displayed:


Thank you!
Secure Tra
You've sent a secure payment of $1.50 USD to
through PayPal. The payment will appe
your card statement as "PAYPAL". You'll receive an email receipt shortly.
View your printable receipt
Return To M

- This page contains button "Return to merchant" pressing which returns user to User Manager payment history page:



- User Manager receives data from PayPal indicating Payment status.

- On a successful payment the appropriate credit is added to user.

PayPal chargeback

When a payment changes status from "Approved" to "Aborted" (For example, "Reversed") User Manager tries to remove credit bought for this money. This is however possible only if the two following requirements are met:

- The credit is not started yet;
- The credit is last for current user, i.e., no other credit is bought after this one.

PayPal payment process description

- The payment data is transmitted directly to PayPal gateway. All operation with money and accounts is processed by PayPal. User Manager knows nothing about it.

- PayPal gateway processes the data and after that sends response to specified User Manager router. It may take time, usually not more than one minute. That means that payment may have status "Started" for a few seconds, the status is updated only when PayPal sends response to User Manager;

- If the option "Secure response" is enabled, secure connection (https) is established between PayPal and User Manager;

- When experiencing problems with HTTPS response from PayPal, "Secure response" may be disabled. Then no certificate will be needed on User Manager router to receive PayPal response;

- Again - PayPal response contains only data required to identify payment in User Manager and detect result status of transaction - was it successful or not. It does not contain any information about the user - credit card number, expiry date or other sensitive data;

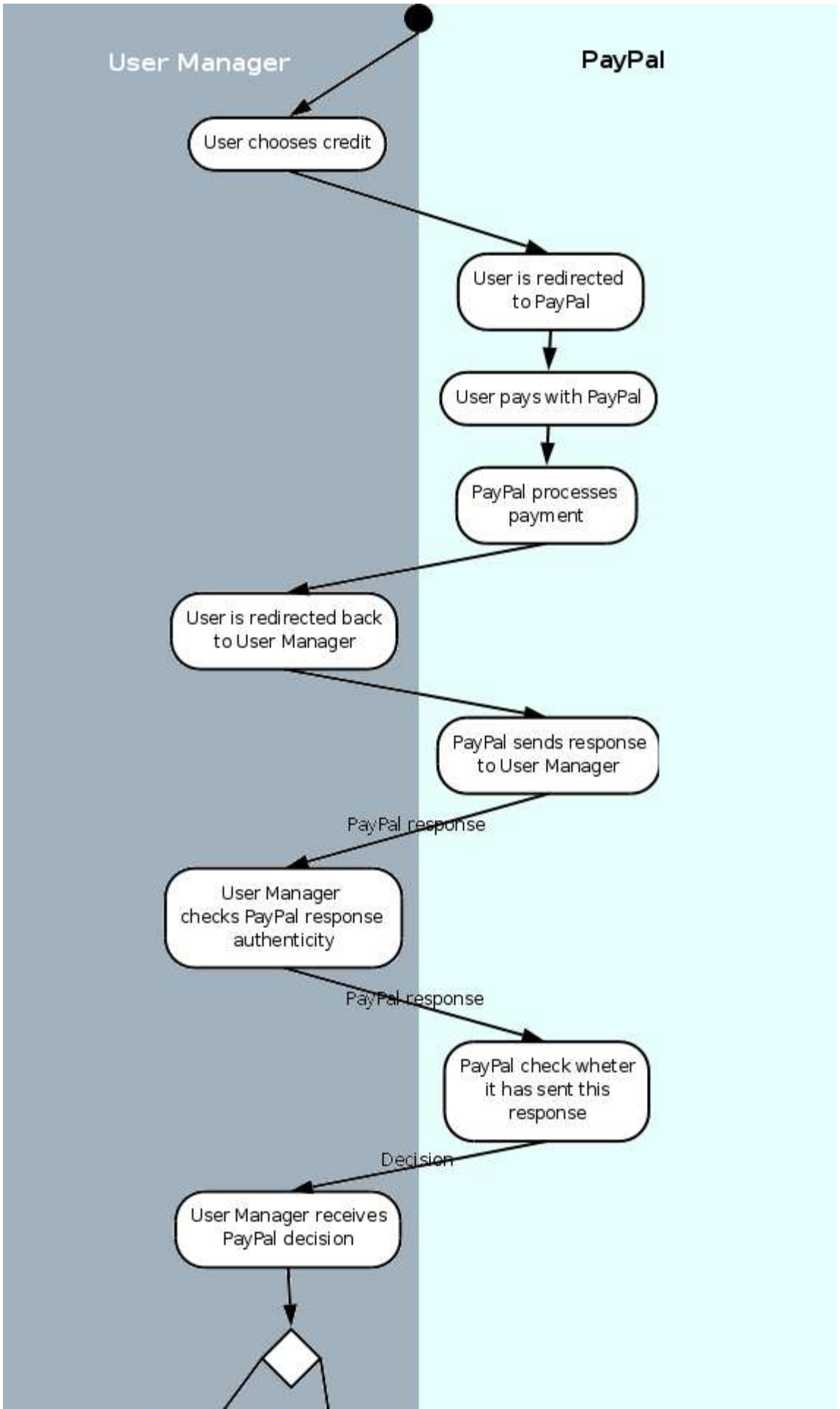
- User Manager sends request to PayPal to verify that this payment response comes from PayPal and not from a hacker. Because of this verification it is not necessary to receive response from PayPal via https - if a Man-In-The-Middle catches data and sends wrong response to User Manager, the verification fails;

- Response verification requires SSL certificate of root certification authority who has signed PayPal certificate. This root CA certificate is imported automatically and can bee seen in certificate section on the router (console or HSConfig);

- User Manager processes the response and updates payment record status;

- If the transaction was successful requested credit is added to user's account;

The payment processing is shown in the following picture:


User Manager
PayPal
User chooses credit
User is redirected to PayPal
User pays with PayPal
PayPal processes payment
User is redirected back to User Manager
PayPal sends response to User Manager
PayPal response
User Manager checks PayPal response authenticity
PayPal response
PayPal check wheter it has sent this response
Decision
User Manager receives PayPal decision

Related activities

HTTPS connection enabling

Creating certificate

Trusted SSL Certificate can be bought from trusted authorities, for example, VeriSign. An unsigned certificate can be generated by hand, using OpenSSL on a Linux box. To do it issue following commands in the shell:

```
openssl genrsa -des3 -out server.key 1024
openssl req -new -key server.key -out server.csr
openssl x509 -req -days 365 -in server.csr -signkey server.key -
out server.crt
```

Two important things:

1. Enter the **same pass phrase** always when asked for "Enter pass phrase for server.key" (Should be 4 times);
2. Enter your **server's domain name**, when asked for **"Common Name (eg, YOUR name) []"**. This is important, because otherwise some browsers may refuse your certificate. For example, if the User Manager server's address is http://userman.mt.lv/userman, then "userman.mt.lv" must be specified as Common Name for the certificate.

After doing this three files will be created:

1. server.crt - Certificate, must be uploaded to router;
2. server.key - Private key, must be uploaded to router;
3. server.csr - Signature request, can/should be deleted;

Upload server.crt and server.key to the router and import them, using the same pass phrase again when asked. **server.crt must be imported before server.key.**

Importing certificate

Certificate file can be then uploaded to the router and imported with command

```
/certificate import file-name=...
```

The command should return

```
    certificates-imported: 1
    private-keys-imported: 1
           files-imported: 1
      decryption-failures: 0
  keys-with-no-certificate: 0
```

If it doesn't, could happen that the file contains private key and certificate sections in incorrect order. In this situation the output should be

```
    certificates-imported: 1
    private-keys-imported: 0
           files-imported: 1
```

```
      decryption-failures: 0
  keys-with-no-certificate: 1
```

Just repeat the same command

```
/certificate import file-name=...
```

once again and the output should be this time

```
    certificates-imported: 0
    private-keys-imported: 1
           files-imported: 1
      decryption-failures: 0
  keys-with-no-certificate: 0
```

Now certificate is imported correctly and ready for use;

Enabling WWW SSL

SSL connections for WWW server can be enabled with command

```
/ip service set www-ssl disabled=no certificate=cert1
```

where cert1 must be replaced by a correct certificate name (from /certificate section)

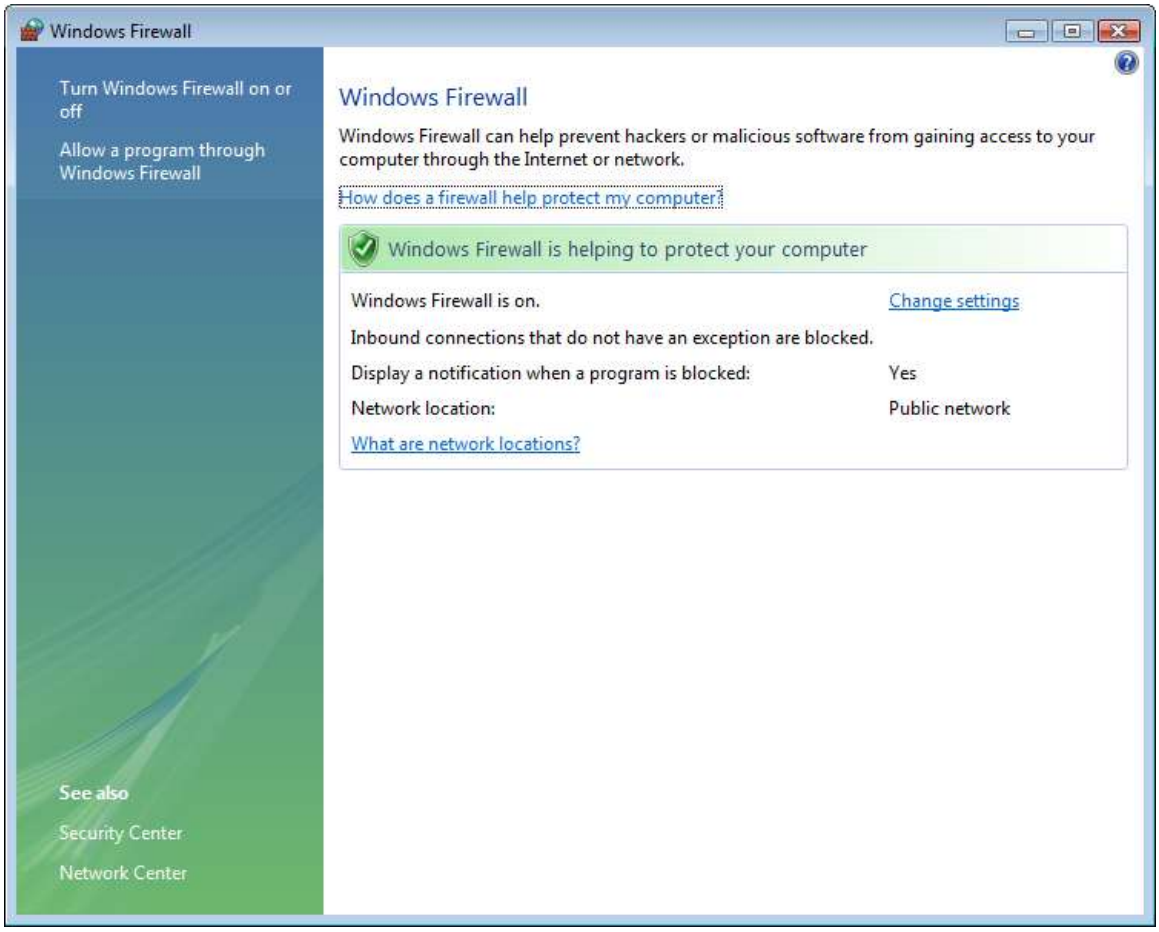
Using Syslogd to Diagnose HauteWRAP Issues

The HauteSpot Syslog Daemon is a simple MS Windows application which allows you to receive and store log messages from any syslog enabled device, in particular the HauteWRAP product family.

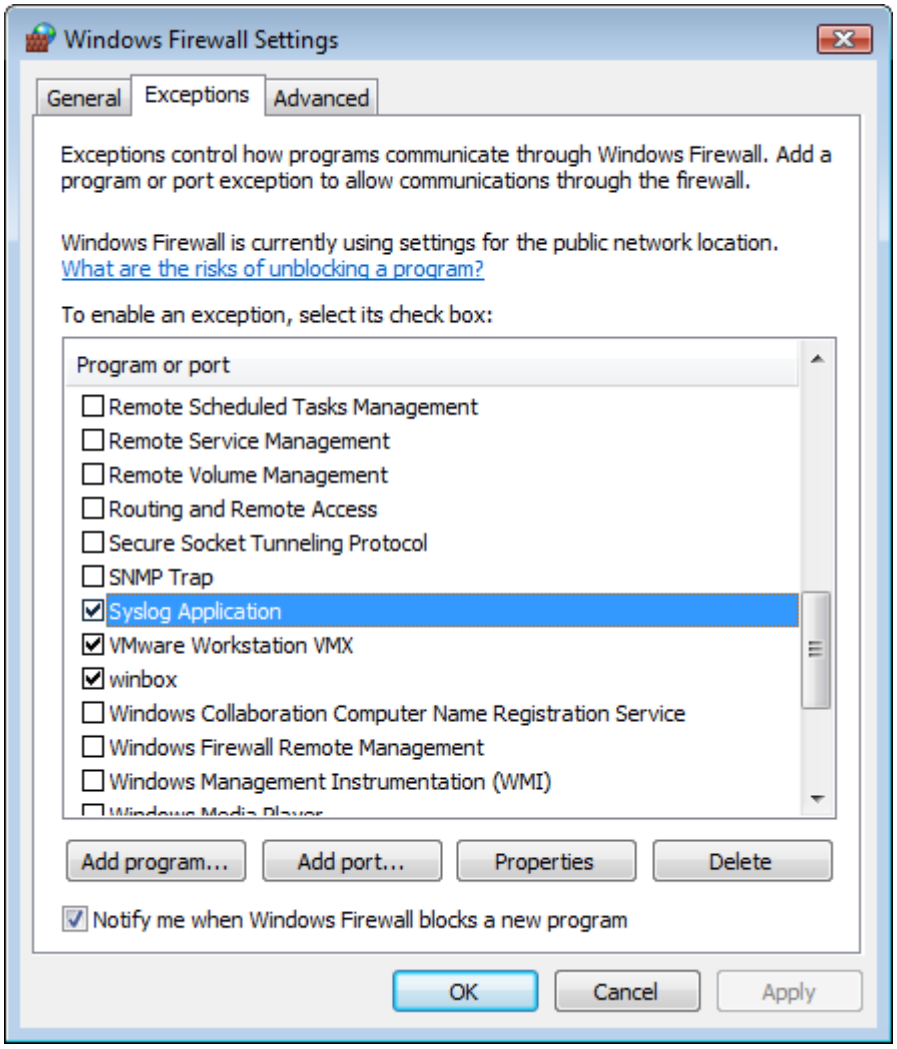
Install and Configure HS Syslogd

First, download and install the HauteSpot Windows Tools which are installed on your HauteWRAPSX or DX product on it's embedded web server. If you have a HauteWRAPCX, LX or HX product, you will need to download the tools from the HauteSpot support web site. See your user manual for details.

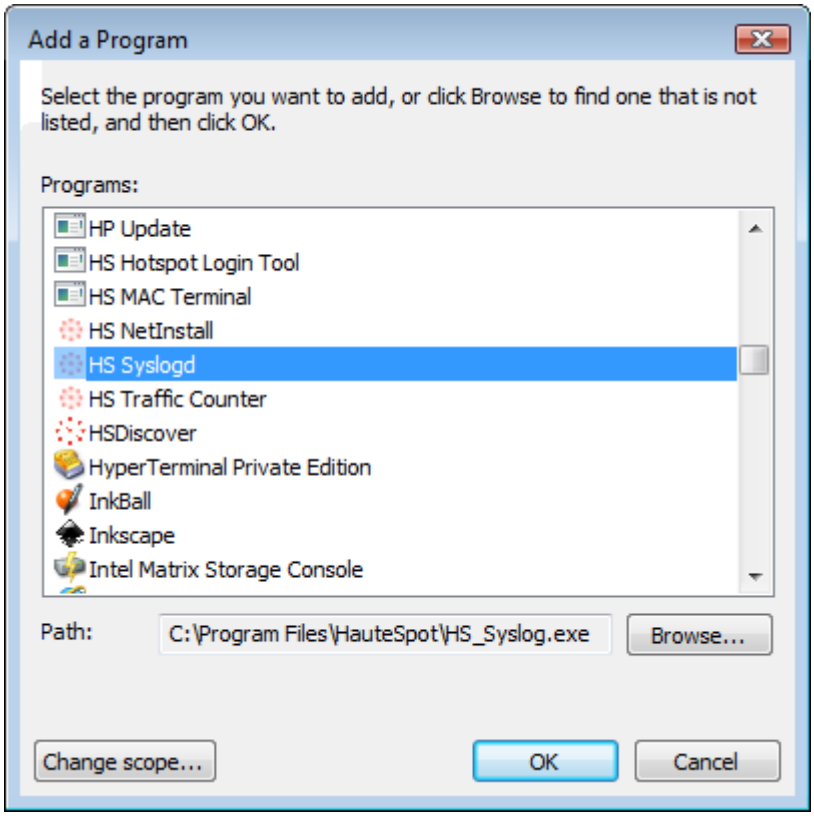
Once installed, in the program group for HauteSpot Networks you will find the application "HS Syslogd". Launch this application in Windows. Make sure that you allow your Windows Firewall application to pass this application through. To do this, go to your Control Panel, Network Connections menu and open Windows Firewall and go to Change Settings.


Windows Firewall
Turn Windows Firewall on or off
Allow a program through Windows Firewall
Windows Firewall
Windows Firewall can help prevent hackers or malicious software from gaining access to your computer through the Internet or network.
How does a firewall help protect my computer?
Windows Firewall is helping to protect your computer
Windows Firewall is on.
Change settings
Inbound connections that do not have an exception are blocked.
Display a notification when a program is blocked:
Yes
Network location:
Public network
What are network locations?
See also
Security Center
Network Center

Then select the exceptions tab and create an exception for Syslogd:



If it is not displayed in the list, click Add program and then scroll the list to HS Syslogd



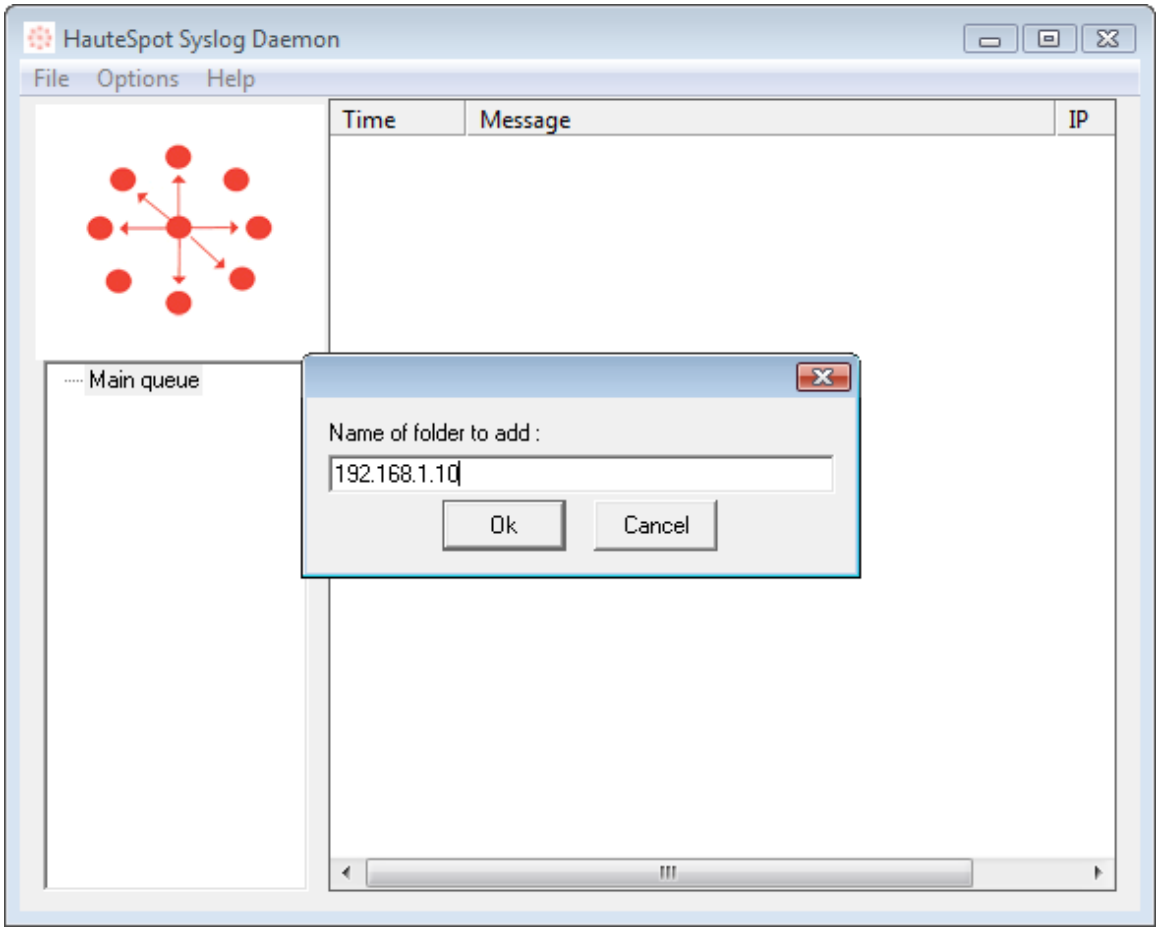
Click OK.

Now if you open the Syslogd application, you will be able to receive messages from remote devices.

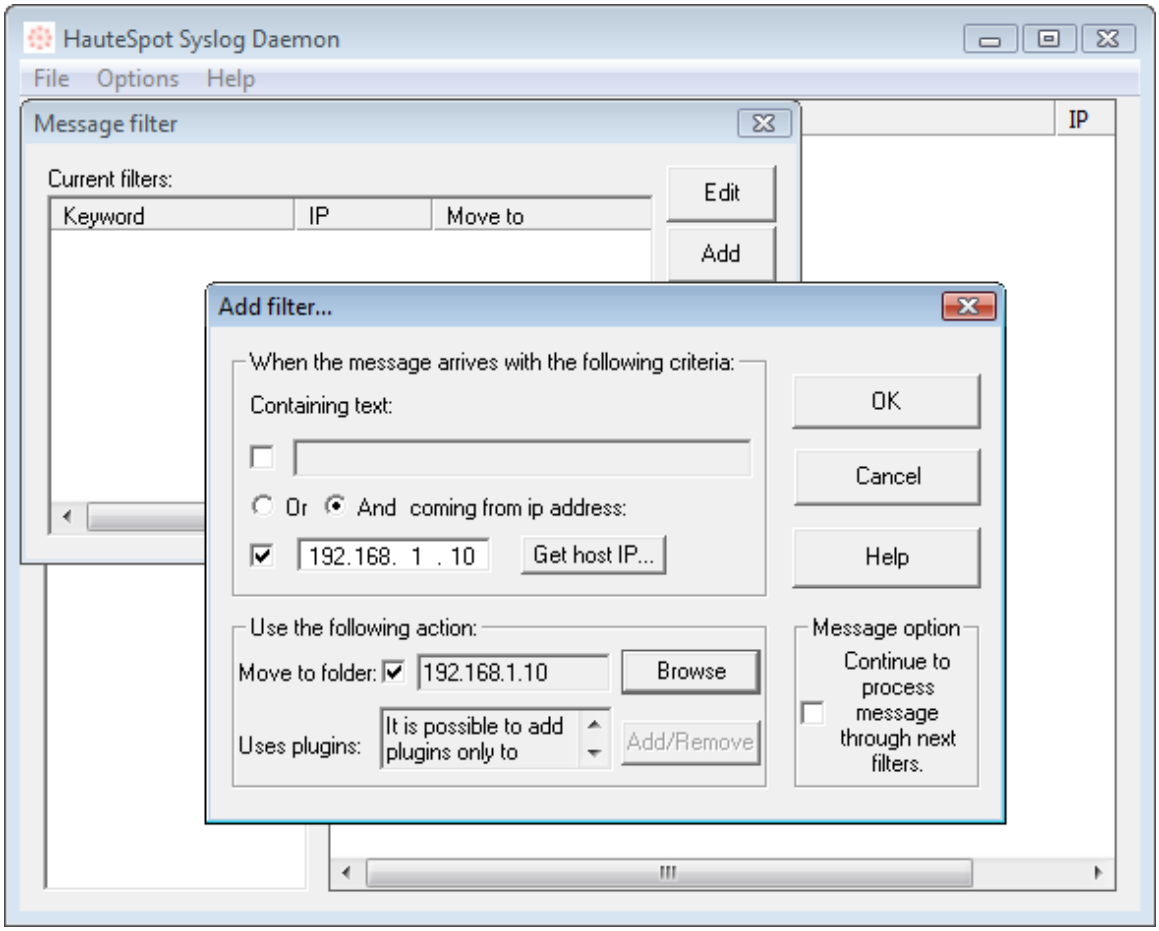
If you minimize the Syslogd application it will remain in your system tray, running in the background.

To better organize your logs, you should create a folder for each device you are monitoring, and then a filter rule that moves incoming messages into that folder as they arrive:

Open the options menu and go to add folder:



Then open options menu and then message filter. Add a filter for the IP address of the device that you are logging, and then browse to the folder that you want incoming log messages from that device to be sorted to.



Now, when messages arrive from that device they will be stored in that folder.

That is it, your log server is now ready to run.

Optional – Run HS Syslogd as a Service

To create a Windows user-defined service, perform the following steps:

Download the Windows Resource Kit from:
http://www.microsoft.com/downloads/details.aspx?familyid=9d467a69-57ff-4ae7-96ee-b18c4790cffd&displaylang=en

You will need the Srvany.exe application and the Instsrv.exe application.

1. At a MS-DOS command prompt(running CMD.EXE), type the following command:

path\INSTSRV.EXE Syslogd path\SRVANY.EXE

where path is the drive and directory of the Windows Resource Kit (i.e., C:\RESKIT) and Syslogd is the name of the service you are creating.

Example:

C:\Program Files\Resource Kit\Instsrv.exe Syslogd C:\Program Files\Resource Kit\Srvany.exe

NOTE: To verify that the service was created correctly, check the registry to verify that the ImagePath value under

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Syslogd
is set to point to SRVANY.EXE. If this is not set correctly, the service will stop shortly after it starts and return an Event ID 7000 "The service name failed to start."

WARNING: Using Registry Editor incorrectly can cause serious problems that may require you to reinstall your operating system. Microsoft cannot guarantee that problems resulting from the incorrect use of Registry Editor can be solved. Use Registry Editor at your own risk.

For information about how to edit the registry, view the "Changing Keys And Values" online Help topic or the "Add and Delete Information in the Registry" and "Edit Registry Data" online Help topics in Registry Editor.

NOTE: You should back up the registry before you edit it.

2. Run Registry Editor (Regedt32.exe)and locate the following subkey:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Syslogd

3. From the Edit menu, click Add Key. Type the following and click OK:

Key Name: Parameters
Class : <leave blank>

4. Select the Parameters key.

5. From the Edit menu, click Add Value. Type the following and click OK:

Value Name: Application
Data Type : REG_SZ
String : <path>\<application.ext>

where <path>\<application.ext> is the drive and full path to the application executable including the extension (i.e., C:\Program Files\HauteSpot\HS_Syslog.exe)

6. Close Registry Editor.

By default, a newly created service it configured to run Automatically when the system is restarted. To change this setting to Manual, run the Services applet from Control Panel and change the Startup value to Manual. A service set to Manual can be started in one of several ways:

- From the Services applet in Control Panel

- From a MS-DOS command prompt, type the following:

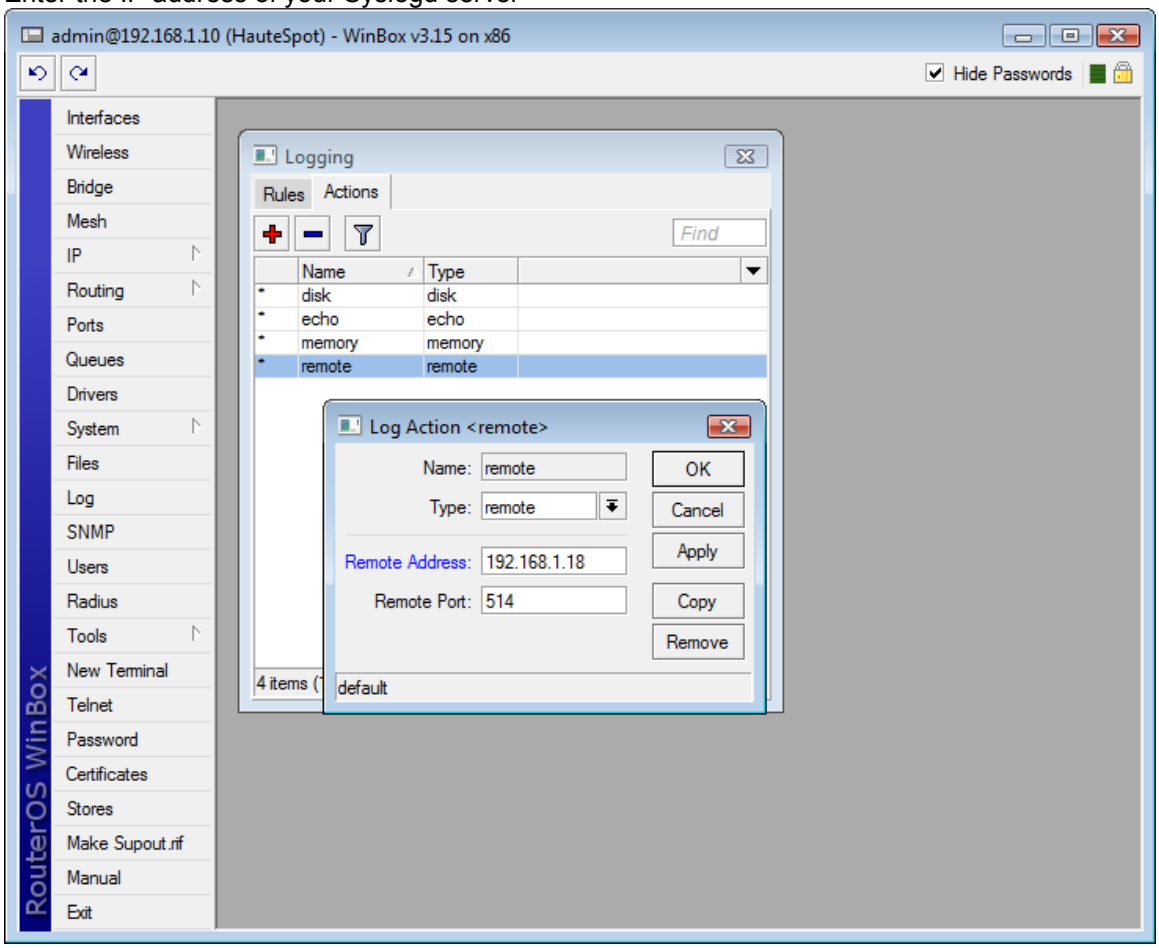
NET START Syslogd

Enable Logging on The HauteWRAP Remote Unit

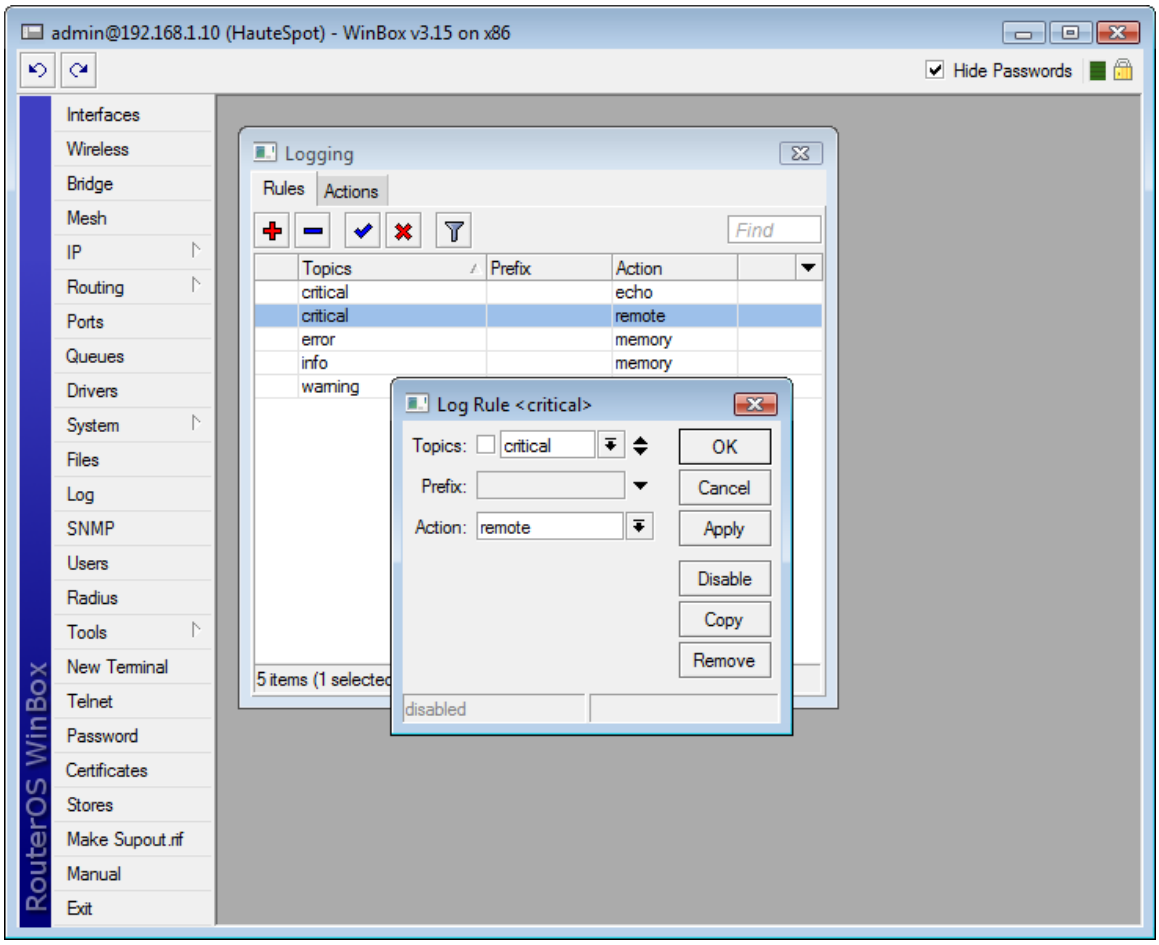
Open a connection to your unit using HSConfig

Open System Logging and go to the Actions Tab, then select the remote line item

Enter the IP address of your Syslogd server

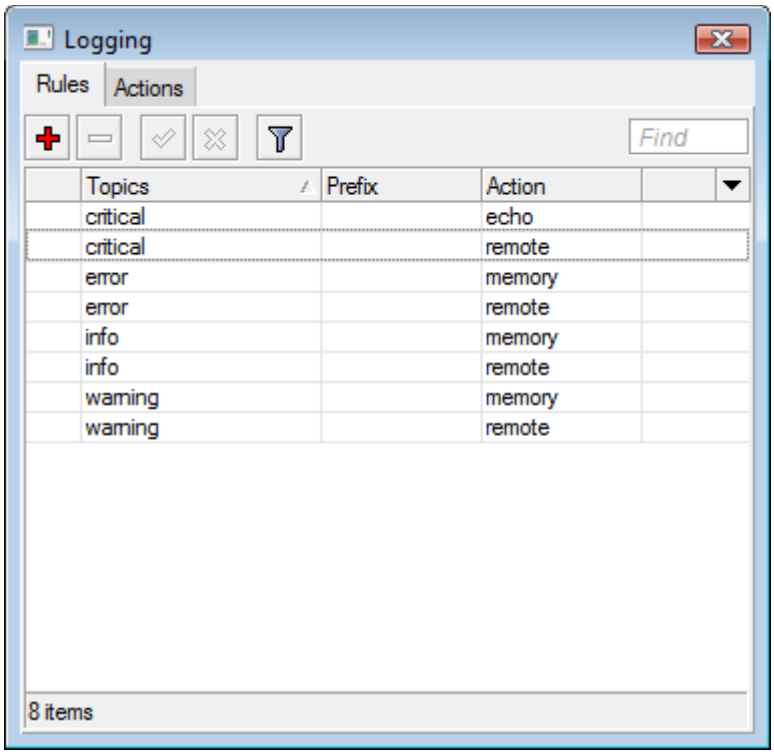


Now go to the Rules tab and click the red + and add a rule for critical messages that will send them to remote



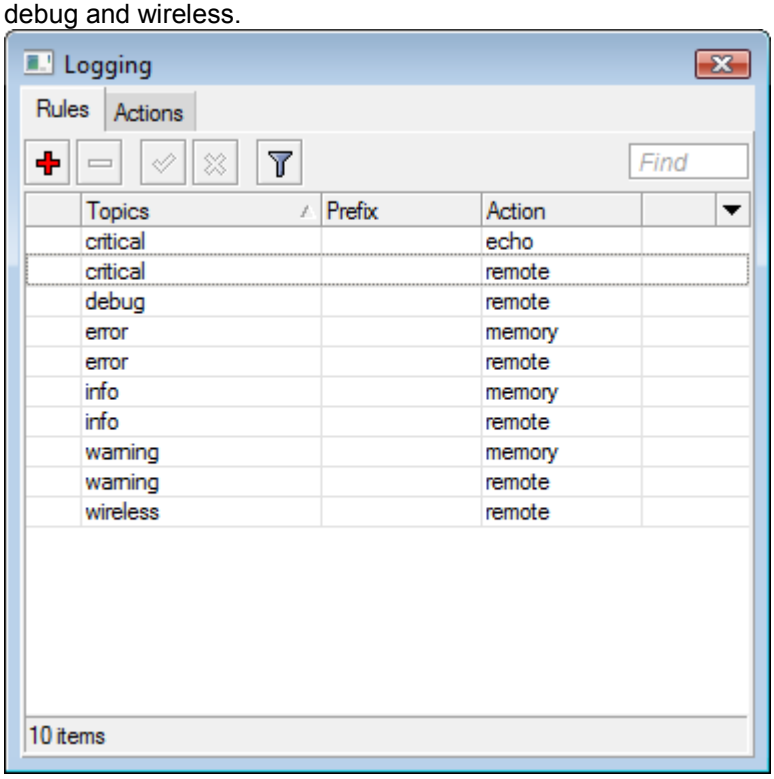
Do the same for error, info and warning message types. This replicates all of the standard logging that will be stored on the remote system, so that these same logs get pushed to your syslogd server.

Your standard list should now look like this:

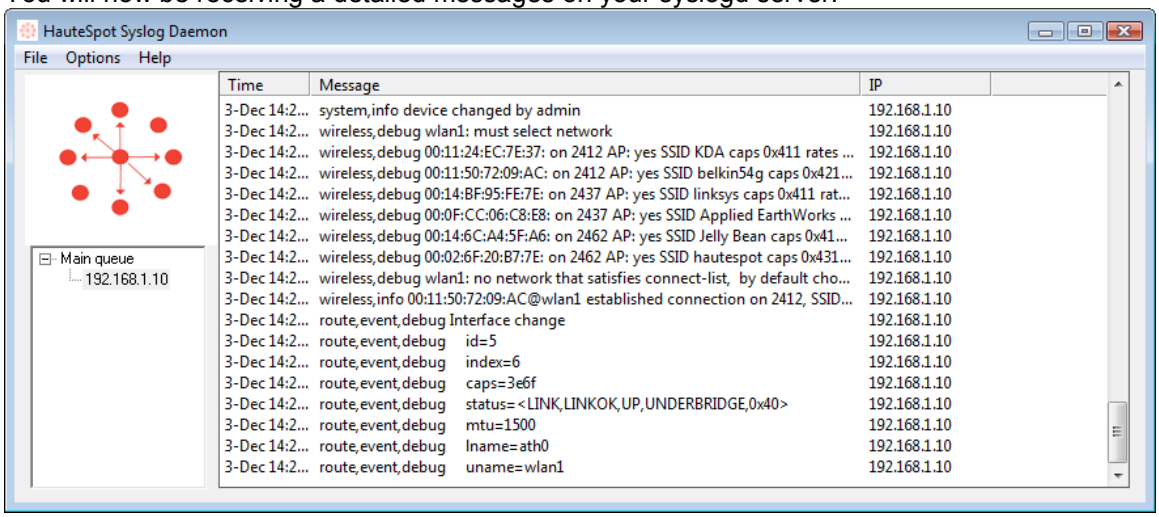


Messages should now be received at your syslogd server.

If you are experiencing system failures, then you should enable more detailed logging. Start with debug and wireless.



You will now be receiving a detailed messages on your syslogd server:


HauteSpot Syslog Daemon
File Options Help
Main queue
192.168.1.10
Time Message IP
3-Dec 14:2... system,info device changed by admin 192.168.1.10
3-Dec 14:2... wireless,debug wlan1: must select network 192.168.1.10
3-Dec 14:2... wireless,debug 00:11:24:EC:7E:37: on 2412 AP: yes SSID KDA caps 0x411 rates ... 192.168.1.10
3-Dec 14:2... wireless,debug 00:11:50:72:09:AC: on 2412 AP: yes SSID belkin54g caps 0x421... 192.168.1.10
3-Dec 14:2... wireless,debug 00:14:BF:95:FE:7E: on 2437 AP: yes SSID linksys caps 0x411 rat... 192.168.1.10
3-Dec 14:2... wireless,debug 00:0F:CC:06:C8:E8: on 2437 AP: yes SSID Applied EarthWorks ... 192.168.1.10
3-Dec 14:2... wireless,debug 00:14:6C:A4:5F:A6: on 2462 AP: yes SSID Jelly Bean caps 0x41... 192.168.1.10
3-Dec 14:2... wireless,debug 00:02:6F:20:B7:7E: on 2462 AP: yes SSID hautespot caps 0x431... 192.168.1.10
3-Dec 14:2... wireless,debug wlan1: no network that satisfies connect-list, by default cho... 192.168.1.10
3-Dec 14:2... wireless,info 00:11:50:72:09:AC@wlan1 established connection on 2412, SSID... 192.168.1.10
3-Dec 14:2... route,event,debug Interface change 192.168.1.10
3-Dec 14:2... route,event,debug id=5 192.168.1.10
3-Dec 14:2... route,event,debug index=6 192.168.1.10
3-Dec 14:2... route,event,debug caps=3e6f 192.168.1.10
3-Dec 14:2... route,event,debug status=<LINK,LINKOK,UP,UNDERBRIDGE,0x40> 192.168.1.10
3-Dec 14:2... route,event,debug mtu=1500 192.168.1.10
3-Dec 14:2... route,event,debug lname=ath0 192.168.1.10
3-Dec 14:2... route,event,debug uname=wlan1 192.168.1.10

The log files are stored in the C;\Program Files\HauteSpot directory as tmplog.log. You can copy and save this log by renaming it after you quit the HS Syslogd application.

Warning: Do not leave robust logging like debug or wireless turned on, as it can adversely effect your network bandwidth availability.

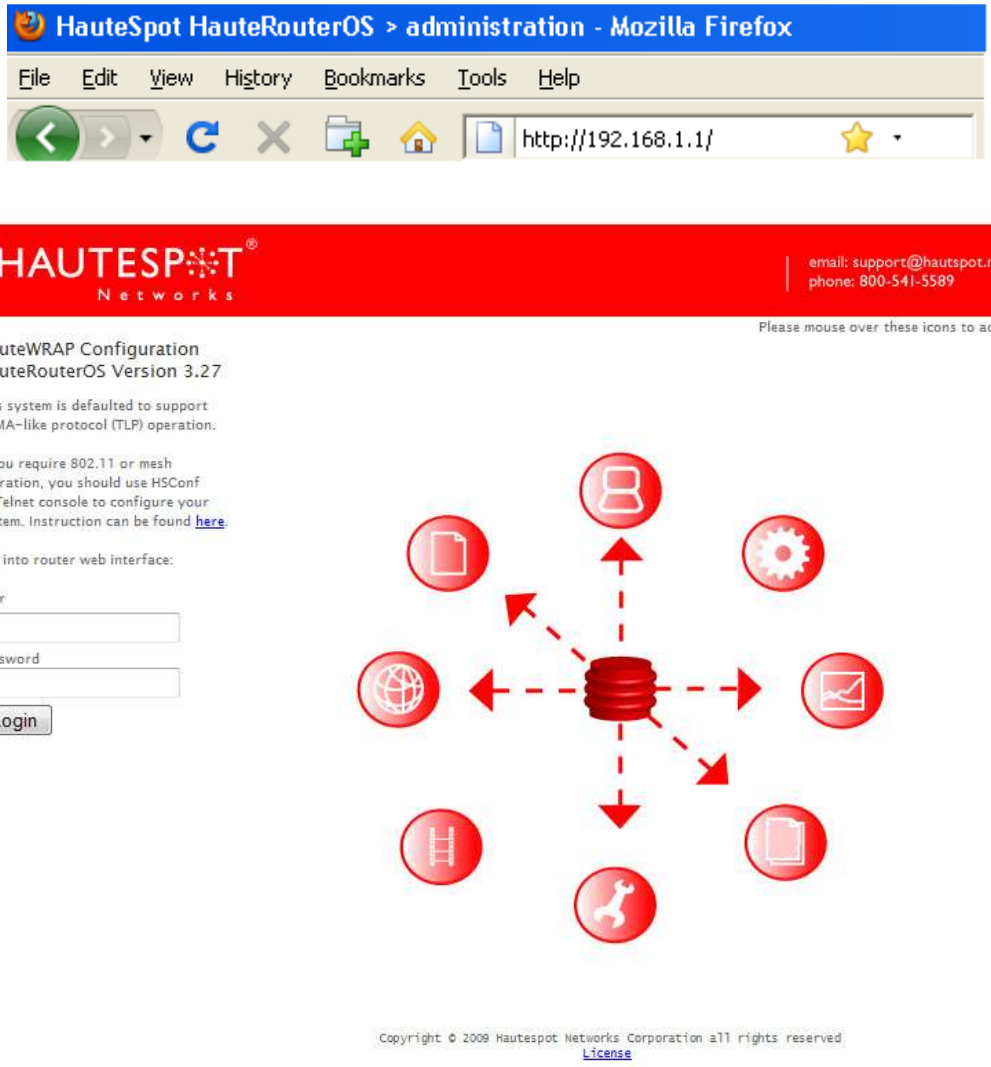
If instructed to do so, you can send your log files to support@hautespot.net for assistance.

Using Webcfg for wireless configuration

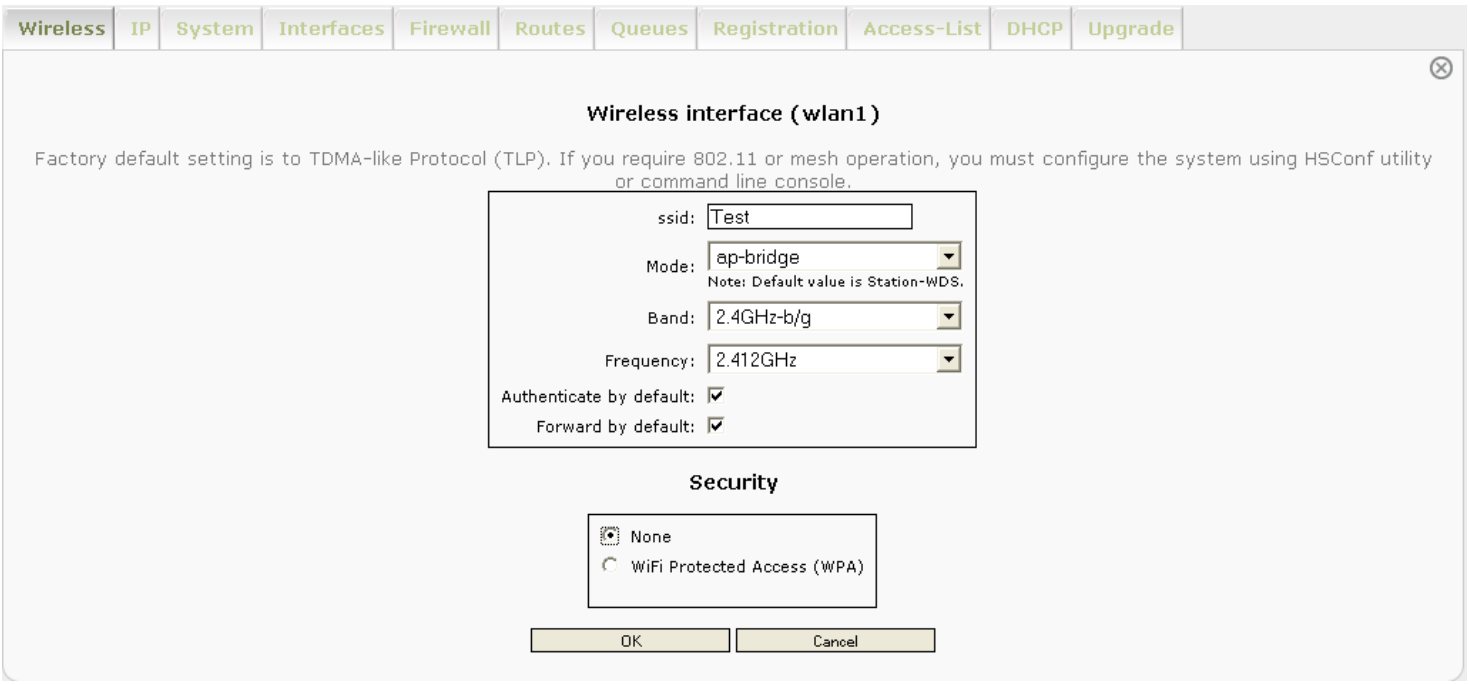
Beginning in the 3.23b revision of firmware the web configuration of the HR-WRAP series was improved for ease of configuration. Many of the features found in the HSConfigurator tool have been removed and advanced users may prefer HSConfigurator instead. The webcfg is intended for installers that are unfamiliar with the HR-WRAP equipment. Using the webcfg the HR-WRAP can be set up in several different operating modes however our guide only will cover ap-bridge and station-wds configuration. In ap-bridge mode the HR-WRAP will broadcast his SSID for other HR-WRAP equipment to see and associate too. By default 802.11 equipment (laptops, PDA's, netbooks etc.) will not be able to associate to the HR-WRAP since it utilizes a proprietary tdma-like protocol. This feature greatly enhances the quality of operation in point to multipoint networks streaming video but can be disabled for 802.11 compliance. Refer to the 'Revert to 802.11' document for instruction. In Station-WDS mode the HR-WRAP will scan all frequencies for an available ap-bridge (using the same SSID).

71 Ensure that your PC is statically addressed to the same network as the HR-WRAP device. By default the HR-WRAP comes addressed at 192.168.1.1. Refer to the 'Configuring Static IP on PC' document for detailed instructions on how to set a static IP address.

72 Open a web browser (Mozilla Firefox, Microsoft Internet Explorer, Google Chrome etc.) and navigate to the IP address of your HR-WRAP to display the login screen. The default login user name is 'admin' with no password. The user name is the upper field and the password is the lower field for the login boxes.


HauteSpot HauteRouterOS > administration - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://192.168.1.1/
HAUTESPOT Networks
email: support@hautspot.net
phone: 800-541-5589
Please mouse over these icons to access features
HauteWRAP Configuration
HauteRouterOS Version 3.27
This system is defaulted to support TDMA-like protocol (TLP) operation.
If you require 802.11 or mesh operation, you should use HSConf or Telnet console to configure your system. Instruction can be found here.
Log into router web interface:
User
Password
Login
Copyright © 2009 Hautespot Networks Corporation all rights reserved
License

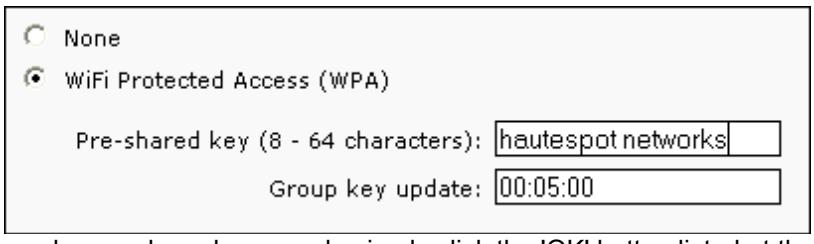
73 Once logged in you will be presented with the Wireless configuration menu. This menu is where most of our changes will be made. The **SSID** field is the broadcast identifier for the network. This is what the devices will use to identify and associate to the ap-bridge. A typical SSID would be 'Security Cameras'. SSID's must match exactly for an association to be made.



By default the **Mode** is set to station-wds for client mode operation. Change to ap-bridge mode for access point operation. When changed to ap-bridge you will notice that you now have the ability to set **Frequency**. Your frequency should have been decided prior to network installation. If a site survey has not been done then we recommend using the first, middle or last frequency available for selection.

The operation **Band** is typically 2.4GHz b/g or 5GHz (depending on antennas). For radios that don't support those band's the default band will work well.

74 Security may be required on some wireless links but is not usually suggested. Security adds overhead to a wireless link that can degrade performance and complicates configuration. By default the HR-WRAP equipment already uses a proprietary protocol that only HR-WRAP equipment can communicate with. If security is used then the Pre-shared key must be configured on all of the clients as well as the access point. They have to match exactly.



75 Once these changes have been made simply click the 'OK' button listed at the bottom of the screen. Associations can be viewed in the Registration menu.

HauteRouterOS Virtual Router Redundancy Protocol

Document revision: 1.4 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

Virtual Router Redundancy Protocol (VRRP) implementation in the HauteRouterOS is RFC2338 compliant. VRRP protocol is used to ensure constant access to some resources. Two or more routers (referred as VRRP Routers in this context) create a highly available cluster (also referred as Virtual routers) with dynamic fail over. Each router can participate in not more than 255 virtual routers per interface. Many modern routers support this protocol.

Network setups with VRRP clusters provide high availability for routers without using clumsy ping-based scripts.

Specifications

Packages required: ***system***
License required: *Level1*
Submenu level: ***/ip vrrp***
Standards and Technologies: VRRP, AH, HMAC-MD5-96 within ESP and AH
Hardware usage: *Not significant*

Description

Virtual Router Redundancy Protocol is an election protocol that provides high availability for routers. A number of routers may participate in one or more virtual routers. One or more IP addresses may be assigned to a virtual router. A node of a virtual router can be in one of the following states:

- **MASTER** state, when the node answers all the requests to the instance's IP addresses. There may only be one MASTER node in a virtual router. This node sends VRRP advertisement packets to all the backup routers (using multicast address) every once in a while (set in **interval** property).
- **BACKUP** state, when the VRRP router monitors the availability and state of the Master Router. It does not answer any requests to the instance's IP addresses. Should master become unavailable (if at least three sequential VRRP packets are lost), election process happens, and new master is

proclaimed based on its priority. For more details on virtual routers, see RFC2338.

VRRP Routers

Submenu level: ***/ip vrrp***

Description

A number of VRRP routers may form a virtual router. The maximal number of clusters on one network is 255 each having a unique VRID (Virtual Router ID). Each router participating in a VRRP cluster must have it priority set to a valid value.

Property Description

name (*name*) - assigned name of the VRRP instance
interface (*name*) - interface name the instance is running on
vrid (*integer*: 0..255; default: **1**) - Virtual Router Identifier (must be unique on one interface)
priority (*integer*: 1..255; default: **100**) - priority of the current node (higher values mean higher priority)
255 - RFC requires that the router that owns the IP addresses assigned to this instance had the priority of 255
interval (*integer*: 1..255; default: **1**) - VRRP update interval in seconds. Defines how frequently the master of the given cluster sends VRRP advertisement packets
preemption-mode (yes | no; default: **yes**) - whether preemption mode is enabled
no - a backup node will not be elected to be a master until the current master fail even if the backup node has higher priority than the current master
yes - the master node always has the priority
authentication (none | simple | ah; default: **none**) - authentication method to use for VRRP advertisement packets
none - no authentication
simple - plain text authentication
ah - Authentication Header using HMAC-MD5-96 algorithm
password (*text*; default: **""**) - password required for authentication depending on method used can be ignored (if no authentication used), 8-character long text string (for plain-text authentication) or 16-character long text string (128-bit key required for AH authentication)
on-backup (*name*; default: **""**) - script to execute when the node switch to backup state
on-master (*name*; default: **""**) - script to execute when the node switch to master state

Notes

All the nodes of one cluster must have the same **vrid**, **interval**, **preemption-mode**, **authentication** and **password**.

As said before, priority of 255 is reserved for the real owner of the virtual router's IP addresses. Theoretically, the owner should have the IP address added statically to its IP address list and also to the VRRP virtual address list, but you should never do this! Any addresses that you are using as virtual addresses (i.e. they are added in **/ip vrrp address**) must not appear in **/ip address** list as they otherwise can cause IP address conflict, which will not be resolved automatically.

Also You must have an IP address (no matter what) on the interface you want to run VRRP on.

To add a VRRP instance on **ether1** interface, forming (because **priority** is **255**) a virtual router with **vrid** of **1**:

```
[admin@Hautespot] ip vrrp> add interface=ether1 vrid=1 priority=255
[admin@Hautespot] ip vrrp> print
Flags: X - disabled, I - invalid, M - master, B - backup
  0  I  name="vr1" interface=ether1 vrid=1 priority=255 interval=1
        preemption-mode=yes authentication=none password="" on-backup=""
        on-master=""

[admin@Hautespot] ip vrrp>
```

Virtual IP addresses

Submenu level: ***/ip vrrp address***

Property Description

address (*IP address*) - IP address belongs to the virtual router
network (*IP address*) - IP address of the network
broadcast (*IP address*) - broadcasting IP address
virtual-router (*name*) - VRRP router's name the address belongs to

Notes

The virtual IP addresses should be the same for each node of a virtual router.

To add a virtual address of **192.168.1.1/24** to the **vr1** VRRP router:

```
[admin@Hautespot] ip vrrp> address add address=192.168.1.1/24 \
\... virtual-router=vr1
[admin@Hautespot] ip vrrp> address print
Flags: X - disabled, A - active
  #    ADDRESS            NETWORK         BROADCAST       VIRUAL-ROUTER
  0    192.168.1.1/24     192.168.1.0     192.168.1.255   vr1

[admin@Hautespot] ip vrrp>
```

A simple example of VRRP fail over

Description

VRRP protocol may be used to make a redundant Internet connection with seamless fail-over. Let us assume that we have 192.168.1.0/24 network and we need to provide highly available Internet connection for it. This network should be NATted (to make fail-over with public IPs, use such dynamic routing protocols as BGP or OSPF together with VRRP). We have connections to two different Internet Service Providers (ISPs), and one of them is preferred (for example, it is cheaper or faster).

This example shows how to configure VRRP on the two routers shown on the diagram. The routers must have initial configuration: interfaces are enabled, each interface have appropriate IP address (note that each of the two interfaces should have an IP address), routing table is set correctly (it should have at least a default route). SRC-NAT or masquerading should also be configured before. See the respective manual chapters on how to make this configuration.

We will assume that the interface the 192.168.1.0/24 network is connected to is named **local** on both VRRP routers

Configuring Master VRRP router

First of all we should create a VRRP instance on this router. We will use the priority of 255 for this router as it should be preferred router.

```
[admin@Hautespot] ip vrrp> add interface=local priority=255
[admin@Hautespot] ip vrrp> print
Flags: X - disabled, I - invalid, M - master, B - backup
  0   M name="vr1" interface=local vrid=1 priority=255 interval=1
        preemption-mode=yes authentication=none password="" on-backup=""
        on-master=""

[admin@Hautespot] ip vrrp>
```

Next the virtual IP address should be added to this VRRP instance

```
[admin@Hautespot] ip vrrp> address add address=192.168.1.1/24 \
\... virtual-router=vr1
[admin@Hautespot] ip vrrp> address print
Flags: X - disabled, A - active
  #    ADDRESS            NETWORK         BROADCAST       VIRTUAL-ROUTER
  0    192.168.1.1/24     192.168.1.0     192.168.1.255   vr1
```

```
[admin@Hautespot] ip vrrp>
```

Now this address should appear in **/ip address** list:

```
[admin@Hautespot] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
  #   ADDRESS            NETWORK         BROADCAST       INTERFACE
  0   10.0.0.1/24        10.0.0.0        10.0.0.255      public
  1   192.168.1.2/24     192.168.1.0     192.168.1.255   local
  2 D 192.168.1.1/24     192.168.1.0     192.168.1.255   local

[admin@Hautespot] ip address>
```

Configuring Backup VRRP router

Now we will create VRRP instance with lower priority (we can use the default value of **100**), so this router will back up the preferred one:

```
[admin@Hautespot] ip vrrp> add interface=local
[admin@Hautespot] ip vrrp> print
Flags: X - disabled, I - invalid, M - master, B - backup
  0   B name="vr1" interface=local vrid=1 priority=100 interval=1
        preemption-mode=yes authentication=none password="" on-backup=""
        on-master=""

[admin@Hautespot] ip vrrp>
```

Now we should add the same virtual address as was added to the master node:

```
[admin@Hautespot] ip vrrp> address add address=192.168.1.1/24 \
\... virtual-router=vr1
[admin@Hautespot] ip vrrp> address print
Flags: X - disabled, A - active
  #    ADDRESS            NETWORK         BROADCAST       VIRTUAL-ROUTER
  0    192.168.1.1/24     192.168.1.0     192.168.1.255   vr1

[admin@Hautespot] ip vrrp>
```

Note that this address will not appear in **/ip address** list:

```
[admin@Hautespot] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
  #   ADDRESS            NETWORK         BROADCAST       INTERFACE
  0   10.1.0.1/24        10.0.0.0        10.0.0.255      public
  1   192.168.1.3/24     192.168.1.0     192.168.1.255   local

[admin@Hautespot] ip address>
```

Testing fail over

Now, when we will disconnect the master router, the backup one will switch to the master state:

```
[admin@Hautespot] ip vrrp> print
Flags: X - disabled, I - invalid, M - master, B - backup
  0   M name="vr1" interface=local vrid=1 priority=100 interval=1
        preemption-mode=yes authentication=none password="" on-backup=""
        on-master=""

[admin@Hautespot] ip vrrp> /ip address print
Flags: X - disabled, I - invalid, D - dynamic
  #    ADDRESS            NETWORK         BROADCAST       INTERFACE
  0    10.1.0.1/24        10.0.0.0        10.0.0.255      public
  1    192.168.1.3/24     192.168.1.0     192.168.1.255   local
  2 D 192.168.1.1/24      192.168.1.0     192.168.1.255   local

[admin@Hautespot] ip vrrp>
```

HauteRouterOS VLAN

Document revision: 1.2 (Tuesday, October 30, 2007)
Applies to: HauteRouterOS V3.0

General Information

Summary

VLAN is an implementation of the 802.1Q VLAN protocol for HauteRouterOS. It allows you to have multiple Virtual LANs on a single ethernet or wireless interface, giving the ability to segregate LANs efficiently. It supports up to 4095 vlan interfaces, each with a unique VLAN ID, per ethernet device. Many routers, including Cisco and Linux based, and many Layer 2 switches also support it.

A VLAN is a logical grouping that allows end users to communicate as if they were physically connected to a single isolated LAN, independent of the physical configuration of the network. VLAN support adds a new dimension of security and cost savings permitting the sharing of a physical network while logically maintaining separation among unrelated users.

Specifications

Packages required: ***system***
License required: *Level1 (limited to 1 vlan)* , *Level3*

Submenu level: ***/interface vlan***
Standards and Technologies: VLAN (IEEE 802.1Q)
Hardware usage: *Not significant*

Description

VLANs are simply a way of grouping a set of switch ports together so that they form a logical network, separate from any other such group. Within a single switch this is straightforward local configuration. When the VLAN extends over more than one switch, the inter-switch links have to become trunks, on which packets are tagged to indicate which VLAN they belong to.

You can use HauteRouterOS (as well as Cisco IOS and Linux) to mark these packets as well as to accept and route marked ones.

As VLAN works on OSI Layer 2, it can be used just as any other network interface without any restrictions. And VLAN successfully passes through Ethernet bridges (for HauteRouterOS bridges you should set **forward-protocols** to **ip**, **arp** and **other**; for other bridges there should be analogical settings).

You can also transport VLANs over wireless links and put multiple VLAN interfaces on a single wireless interface. Note that as VLAN is not a full tunnel protocol (i.e., it does not have additional fields to transport MAC addresses of sender and recipient), the same limitation applies to bridging over VLAN as to bridging plain wireless interfaces. In other words, while wireless clients may participate in VLANs put on wireless interfaces, it is not possible to have VLAN put on a wireless interface in station mode bridged with any other interface.

Currently supported Ethernet interfaces

This is a list of network interfaces on which VLAN was tested and worked. Note that there might be many other interfaces that support VLAN, but they just were not checked.

- Realtek 8139
- Intel PRO/100
- Intel PRO1000 server adapter
- National Semiconductor DP83816 based cards (HauteWRAP200 onboard Ethernet, HauteWRAP 24 card)
- National Semiconductor DP83815 (Soekris onboard Ethernet)
- VIA VT6105M based cards (HauteWRAP 44 card)
- VIA VT6105
- VIA VT6102 (VIA EPIA onboard Ethernet)

This is a list of network interfaces on which VLAN was tested and worked, but **WITHOUT LARGE PACKET (>1496 bytes) SUPPORT**:

- 3Com 3c59x PCI
- DEC 21140 (tulip)

Additional Resources

- http://www.csd.uwo.ca/courses/CS457a/reports/handin/jpbojtos/A2/trunking.htm
- http://www.cisco.com/univercd/cc/td/doc/product/software/ios121/121newft/121t/121t3/dtbridge.htm#xtocid114533
- http://www.cisco.com/warp/public/473/27.html#tagging
- http://www.cisco.com/warp/public/538/7.html
- http://www.nwfusion.com/news/tech/2001/0305tech.html
- http://www.intel.com/network/connectivity/resources/doc_library/tech_brief/virtual_lans.htm

VLAN Setup

Submenu level: ***/interface vlan***

Property Description

arp (disabled | enabled | proxy-arp | reply-only; default: **enabled**) - Address Resolution Protocol setting
disabled - the interface will not use ARP protocol
enabled - the interface will use ARP protocol
proxy-arp - the interface will be an ARP proxy
reply-only - the interface will only reply to the requests originated to its own IP addresses, but neighbor MAC addresses will be gathered from **/ip arp** statically set table only
interface (*name*) - physical interface to the network where are VLANs
mtu (*integer*; default: **1500**) - Maximum Transmission Unit
name (*name*) - interface name for reference
vlan-id (*integer*; default: **1**) - Virtual LAN identifier or tag that is used to distinguish VLANs. Must be equal for all computers in one VLAN.

Notes

MTU should be set to 1500 bytes as on Ethernet interfaces. But this may not work with some Ethernet cards that do not support receiving/transmitting of full size Ethernet packets with VLAN header added (1500 bytes data + 4 bytes VLAN header + 14 bytes Ethernet header). In this situation MTU 1496 can be used, but note that this will cause packet fragmentation if larger packets have to be sent over interface. At the same time remember that MTU 1496 may cause problems if path MTU discovery is not working properly between source and destination.

Example

To add and enable a VLAN interface named **test** with **vlan-id**=1 on interface **ether1**:

```
[admin@Hautespot] interface vlan> add name=test vlan-id=1 interface=ether1
[admin@Hautespot] interface vlan> print
Flags: X - disabled, R - running
```

```
  #    NAME                MTU  ARP       VLAN-ID INTERFACE
  0 X  test                1500 enabled   1       ether1
[admin@Hautespot] interface vlan> enable 0
[admin@Hautespot] interface vlan> print
Flags: X - disabled, R - running
  #    NAME                MTU  ARP       VLAN-ID INTERFACE
  0  R test                1500 enabled   1       ether1
[admin@Hautespot] interface vlan>
```

Application Example

VLAN example on Hautespot Routers

Let us assume that we have two or more HauteRouterOS routers connected with a hub. Interfaces to the physical network, where VLAN is to be created is **ether1** for all of them (it is needed only for example simplification, it is NOT a must).

To connect computers through VLAN they must be connected physically and unique IP addresses should be assigned them so that they could ping each other. Then on each of them the VLAN interface should be created:

```
[admin@Hautespot] interface vlan> add name=test vlan-id=32 interface=ether1
[admin@Hautespot] interface vlan> print
Flags: X - disabled, R - running
  #    NAME                MTU  ARP       VLAN-ID INTERFACE
  0  R test                1500 enabled   32      ether1
[admin@Hautespot] interface vlan>
```

If the interfaces were successfully created, both of them will be **running**. If computers are connected incorrectly (through network device that does not retransmit or forward VLAN packets), either both or one of the interfaces will not be **running**.

When the interface is running, IP addresses can be assigned to the VLAN interfaces.

On the Router 1:

```
[admin@Hautespot] ip address> add address=10.10.10.1/24 interface=test
[admin@Hautespot] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
  #   ADDRESS            NETWORK         BROADCAST       INTERFACE
  0   10.0.0.204/24      10.0.0.0        10.0.0.255      ether1
  1   10.20.0.1/24       10.20.0.0       10.20.0.255     pc1
  2   10.10.10.1/24      10.10.10.0      10.10.10.255    test
[admin@Hautespot] ip address>
```

On the Router 2:

```
[admin@Hautespot] ip address> add address=10.10.10.2/24 interface=test
[admin@Hautespot] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
  #   ADDRESS            NETWORK         BROADCAST       INTERFACE
  0   10.0.0.201/24      10.0.0.0        10.0.0.255      ether1
  1   10.10.10.2/24      10.10.10.0      10.10.10.255    test
[admin@Hautespot] ip address>
```

If it set up correctly, then it is possible to ping Router 2 from Router 1 and vice versa:

```
[admin@Hautespot] ip address> /ping 10.10.10.1
10.10.10.1 64 byte pong: ttl=255 time=3 ms
10.10.10.1 64 byte pong: ttl=255 time=4 ms
10.10.10.1 64 byte pong: ttl=255 time=10 ms
10.10.10.1 64 byte pong: ttl=255 time=5 ms
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 3/10.5/10 ms
[admin@Hautespot] ip address> /ping 10.10.10.2
10.10.10.2 64 byte pong: ttl=255 time=10 ms
10.10.10.2 64 byte pong: ttl=255 time=11 ms
10.10.10.2 64 byte pong: ttl=255 time=10 ms
10.10.10.2 64 byte pong: ttl=255 time=13 ms
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 10/11/13 ms
[admin@Hautespot] ip address>
```

HauteRouterOS Web Proxy

Document revision: 1.1 (Tuesday, October 30, 2007)

Applies to: HauteRouterOS V3.0

General Information

Summary

The HauteRouterOS implements the following proxy server features:

- Regular HTTP proxy
- Transparent proxy. Can be transparent and regular at the same time
- Access list by source, destination, URL and requested method
- Cache access list (specifies which objects to cache, and which not)
- Direct Access List (specifies which resources should be accessed directly, and which - through another proxy server)
- Logging facility

Quick Setup Guide

To set up a 1 GiB large web cache which will listen on port 8000, do the following:

```
[admin@Hautespot] ip web-proxy> set enabled=yes port=8000 max-cache-
size=1048576
[admin@Hautespot] ip web-proxy> print
                   enabled: yes
               src-address: 0.0.0.0
                      port: 8000
                  hostname: proxy
         transparent-proxy: no
              parent-proxy: 0.0.0.0:0
       cache-administrator: webmaster
           max-object-size: 4096 KiB
               cache-drive: system
            max-cache-size: 1048576 KiB
        max-ram-cache-size: unlimited
                    status: rebuilding-cache
        reserved-for-cache: 9216 KiB
    reserved-for-ram-cache: 2048 KiB
[admin@Hautespot] ip web-proxy>
```

Remember to secure your proxy by preventing unauthorized access to it.

Specifications

Packages required: ***web-proxy***
License required: *Level3*
Submenu level: ***/ip web-proxy***
Standards and Technologies: HTTP/1.0, HTTP/1.1, FTP
Hardware usage: *uses memory and disk space, if available (see description below)*

Description

Web proxy performs Internet object cache function by storing requested Internet objects, i.e., data available via HTTP and FTP protocols on a system positioned closer to the recipient than the site the data is originated from. Here 'closer' means increased path reliability, speed or both. Web browsers can then use the local proxy cache to speed up access and reduce bandwidth consumption.

When setting up Web proxy, make sure it serves only your clients, and is not misused as relay. Please read the security notice in the Access List Section!

Note that it may be useful to have Web proxy running even with no cache when you want to use it as something like HTTP and FTP firewall (for example, denying access to mp3 files) or to redirect requests to external proxy transparently.

Setup

Submenu level: ***/ip web-proxy***

Property Description

cache-administrator (*text*; default: **webmaster**) - administrator's e-mail displayed on proxy error page
cache-drive (system | *name*; default: **system**) - specifies the target disk drive to be used for storing cached objects. You can use shell expansion to see available drives
enabled (yes | no; default: **no**) - specifies whether the web proxy is enabled
hostname (*text*; default: **proxy**) - hostname (DNS or IP address) of the web proxy
max-cache-size (none | unlimited | *integer*: 0..4294967295; default: **none**) - specifies the maximal disk cache size, measured in kibibytes
max-object-size (*integer*; default: **4096**) - objects larger than the size specified will not be saved on disk. The value is measured in kibibytes. If you wish to get a high bytes hit ratio, you should probably increase this (one 2 MiB object hit counts for 2048 1KiB hits). If you wish to increase speed more than your want to save bandwidth you should leave this low
max-ram-cache-size (none | unlimited | *integer*: 0..4294967295; default: **unlimited**) - specifies the maximal memory cache size, measured in kibibytes
parent-proxy (*IP address*:*port*; default: **0.0.0.0:0**) - specifies upper-level (parent) proxy
port (*port*{1,10}; default: **3128**) - specifies the port(s) the web proxy will be listening on
reserved-for-cache (*read-only: integer*; default: **0**) - specifies allocated memory cache size, measured in kibibytes
reserved-for-ram-cache (*read-only: integer*; default: **2048**) - specifies allocated memory cache size, measured in kibibytes
src-address (*IP address*; default: **0.0.0.0**) - the web-proxy will use this address connecting to the parent proxy or web site.
0.0.0.0 - appropriate **src-address** will be automatically taken from the routing table
status (*read-only: text*; default: **stopped**) - display status information of the proxy server
stopped - proxy is disabled and is not running
rebuilding-cache - proxy is enabled and running, existing cache is being verified
running - proxy is enabled and running
stopping - proxy is shutting down (max 10s)
clearing-cache - proxy is stopped, cache files are being removed
creating-cache - proxy is stopped, cache directory structure is being created
dns-missing - proxy is enabled, but not running because of unknown DNS server (you should specify it under **/ip dns**)
invalid-address - proxy is enabled, but not running because of invalid address (you should change address or port)
invalid-cache-administrator - proxy is enabled, but not running because of invalid cache-administrator's e-mail address
invalid-hostname - proxy is enabled, but not running because of invalid hostname (you should set a valid hostname value)
error-logged - proxy is not running because of unknown error. This error is logged as System-Error. Please, send us this error and some description, how it happened
reserved-for-cache (integer) - maximal cache size, that is accessible to web-proxy
transparent-proxy (yes | no; default: **no**) - specifies whether the proxy uses transparent mode or not

Notes

By default the proxy cache can use as much disk space as there is allocated for it. When the system allocates the space for the proxy cache, 1/7th of the total partition (disk) size is reserved for the system, but not less than 50MB. The rest is left for the proxy cache. The system RAM size is considered as well when allocating the cache size. The cache size is limited so, that there are at least 15MB of RAM per 1GB of cache plus 55MB of RAM is reserved for the system. **max-cache-size** is also taken in account, so the cache will not occupy more than it is specified in this property. The effective limit is calculated as a minimum of all three limits. Note also that RouterOS supports up to 950MB of memory.

Considering the previous note, you should be aware that you will not be able to enable web proxy, if you have less than 60MB of RAM on your router

Expire time of cache entries can be different for each HTML page (specified in headers). But, if there is no such header, the entry will be considered fresh for not more than 72 hours.

The web proxy listens to all IP addresses that the router has in its IP address list.

Example

To enable the proxy on port 8080:

```
[admin@Hautespot] ip web-proxy> set enabled=yes port=8080
[admin@Hautespot] ip web-proxy> print
                   enabled: yes
               src-address: 0.0.0.0
                      port: 8080
                  hostname: proxy
         transparent-proxy: no
              parent-proxy: 0.0.0.0:0
       cache-administrator: webmaster
           max-object-size: 4096 KiB
               cache-drive: system
            max-cache-size: none
        max-ram-cache-size: unlimited
                    status: running
        reserved-for-cache: 0 KiB
    reserved-for-ram-cache: 2048 KiB
[admin@Hautespot] ip web-proxy>
```

Access List

Submenu level: ***/ip web-proxy access***

Description

Access list is implemented in the same way as Hautespot RouterOS firewall rules. Rules are processed from the top to the bottom. First matching rule specifies decision of what to do with this connection. There is a total of 6 classifiers that specify matching constraints. If none of these classifiers is specified, the particular rule will match every connection.

If connection is matched by a rule, **action** property of this rule specifies whether connection will be allowed or not. If the particular connection does not match any rule, it will be allowed.

By default, there exists one rule which prevents **connect** requests to ports other then **443** and **563**.

Property Description

action (allow | deny; default: **allow**) - specifies whether to pass or deny matched packets
dst-address (*IP address/netmask*) - destination address of the IP packet
dst-port (*port*{1,10}) - a list or range of ports the packet is destined to
local-port (*port*) - specifies the port of the web proxy via which the packet was received. This value should match one of the ports web proxy is listening on.
method (any | connect | delete | get | head | options | post | put | trace) - HTTP method used in the request (see HTTP Methods section in the end of this document)
src-address (*IP address/netmask*) - source address of the IP packet
url (*wildcard*) - the URL of the HTTP request

Notes

There is one rule by default, that disallows **connect** method connections to ports other than **443** (https) and **563** (snews). **connect** method is a security hole that allows connections (transparent tunneling) to any computer using any protocol. It is used mostly by spammers, as they found it very convenient to use others' mail (SMTP) servers as anonymous mail relay to send spam over the Internet.

It is strongly recommended to deny all IP addresses except those behind the router as the proxy still may be used to access your internal-use-only (intranet) web servers. Also, consult examples in Firewall Manual on how to protect your router.

Wildcard property **url** matches a complete string (i.e., they will not match "example.com" if they are set to "example"). Available wildcards are '*' (match any number of any characters) and '?' (match any one character). Regular expressions are also accepted here, but if the property should be treated as a regular expression, it should start with a colon (':').

Small hits in using regular expressions:

- **** symbol sequence is used to enter **** character in console

- **\.** pattern means **.** only (in regular expressions single dot in pattern means any symbol)
- to show that no symbols are allowed before the given pattern, we use **^** symbol at the beginning of the pattern
- to specify that no symbols are allowed after the given pattern, we use **$** symbol at the end of the pattern
- to enter **[** or **]** symbols, you should escape them with backslash ****.

Example

The default rule:

```
[admin@Hautespot] ip web-proxy access> print
Flags: X - disabled, I - invalid
 0   ;;; allow CONNECT only to SSL ports 443 [https] and 563 [snews]
     dst-port=!443,563 method=connect action=deny
[admin@Hautespot] ip web-proxy access>
```

To disallow download of .MP3 and .MPG files and FTP connections other than from the **10.0.0.1** server:

```
[admin@Hautespot] ip web-proxy access> add url=":\\.mp\[3g\]$" action=deny
[admin@Hautespot] ip web-proxy access> add src-address=10.0.0.1/32
action=allow
[admin@Hautespot] ip web-proxy access> add url="ftp://*" action=deny
[admin@Hautespot] ip web-proxy access> print
Flags: X - disabled, I - invalid
 0   ;;; allow CONNECT only to SSL ports 443 [https] and 563 [snews]
     dst-port=!443,563 method=connect action=deny

 1   url=":\.mp[3g]$" action=deny

 2   src-address=10.0.0.1/32 action=allow

 3   url="ftp://*" action=deny
[admin@Hautespot] ip web-proxy access>
```

Direct Access List

Submenu level: ***/ip web-proxy direct***

Description

If **parent-proxy** property is specified, it is possible to tell proxy server whether to try to pass the request to the parent proxy or to resolve it connecting to the requested server directly. Direct Access List is managed just like Proxy Access List described in the previous chapter except the **action** argument.

Property Description

action (allow | deny; default: **allow**) - specifies the action to perform on matched packets
allow - always resolve matched requests directly bypassing the parent router
deny - resolve matched requests through the parent proxy. If no one is specified this has the same effect as **allow**
dst-address (*IP address/netmask*) - destination address of the IP packet
dst-port (*port*{1,10}) - a list or range of ports the packet is destined to
local-port (*port*) - specifies the port of the web proxy via which the packet was received. This value should match one of the ports web proxy is listening on.
method (any | connect | delete | get | head | options | post | put | trace) - HTTP method used in the request (see HTTP Methods section in the end of this document)
src-address (*IP address/netmask*) - source address of the IP packet
url (*wildcard*) - the URL of the HTTP request

Notes

Unlike the access list, the direct proxy access list has default action equal to **deny**. It takes place when no rules are specified or a particular request did not match any rule.

Cache Management

Submenu level: ***/ip web-proxy cache***

Description

Cache access list specifies, which requests (domains, servers, pages) have to be cached locally by web proxy, and which not. This list is implemented exactly the same way as web proxy access list. Default action is to cache object (if no matching rule is found).

Property Description

action (allow | deny; default: **allow**) - specifies the action to perform on matched packets
allow - cache objects from matched request
deny - do not cache objects from matched request
dst-address (*IP address/netmask*) - destination address of the IP packet
dst-port (*port*{1,10}) - a list or range of ports the packet is destined to
local-port (*port*) - specifies the port of the web proxy via which the packet was received. This value should match one of the ports web proxy is listening on.
method (any | connect | delete | get | head | options | post | put | trace) - HTTP method used in the request (see HTTP Methods section in the end of this document)
src-address (*IP address/netmask*) - source address of the IP packet
url (*wildcard*) - the URL of the HTTP request

Complementary Tools

Description

Web proxy has additional commands to handle non-system drive used for caching purposes and to recover the proxy from severe file system errors.

Command Description

check-drive - checks non-system cache drive for errors
clear-cache - deletes existing cache and creates new cache directories
format-drive - formats non-system cache drive and prepairs it for holding the cache

Transparent Mode

Description

Transparent proxy feature performs request caching invisibly to the end-user. This way the user does not notice that his connection is being processed by the proxy and therefore does not need to perform any additional configuration of the software he is using.

Transparent proxy combined with bridge greatly simplifies deployment of web proxy in the existing infrastructure.

To enable the transparent mode, place a firewall rule in destination NAT, specifying which connections, *id est* traffic coming to which ports should be redirected to the proxy.

Notes

Only HTTP traffic is supported in transparent mode of the web proxy. HTTPS and FTP protocols are not going to work this way.

Example

To configure the router to transparently redirect all connections coming from **ether1** interface to port **80** to the web proxy listening on port **8080**, then add the following destination NAT rule:

```
[admin@Hautespot] > /ip firewall nat add in-interface=ether1 dst-port=80 \
\... protocol=tcp action=redirect to-ports=8080 chain=dstnat
[admin@Hautespot] > /ip firewall nat print
Flags: X - disabled, I - invalid, D - dynamic
 0   chain=dstnat protocol=tcp in-interface=ether1 dst-port=80 action=redirect
```

```
     to-ports=8080
[admin@Hautespot] >
```

Notes

Be aware, that you will not be able to access the router's web page after addition of the rule above unless you will change the port for the **www** service under **/ip service** submenu to a different value or explicitly exclude router's IP address from those to be matched, like:

```
 /ip firewall nat add in-interface=ether1 dst-port=80 \
\... protocol=tcp action=redirect to-ports=8080 chain=dstnat dst-
address=!1.1.1.1/32
```

It is assumed that the router's address is **1.1.1.1/32**.

HTTP Methods

Description

OPTIONS

This method is a request of information about the communication options available on the chain between the client and the server identified by the **Request-URI**. The method allows the client to determine the options and (or) the requirements associated with a resource without initiating any resource retrieval

GET

This method retrieves whatever information identified by the **Request-URI**. If the **Request-URI** refers to a data processing process than the response to the **GET** method should contain data produced by the process, not the source code of the process procedure(-s), unless the source is the result of the process.

The **GET** method can become a *conditional* **GET** if the request message includes an **If-Modified-Since, If-Unmodified-Since, If-Match, If-None-Match**, or **If-Range** header field. The conditional **GET** method is used to reduce the network traffic specifying that the transfer of the entity should occur only under circumstances described by conditional header field(-s).

The **GET** method can become a *partial* **GET** if the request message includes a **Range** header field. The partial **GET** method intends to reduce unnecessary network usage by requesting only parts of entities without transferring data already held by client.

The response to a **GET** request is cacheable if and only if it meets the requirements for HTTP caching.

HEAD

This method shares all features of **GET** method except that the server must not return a message-body in the response. This retrieves the metainformation of the entity implied by the request which leads to a wide usage of it for testing hypertext links for validity, accessibility, and recent modification.

The response to a **HEAD** request may be cacheable in the way that the information contained in the response may be used to update previously cached entity identified by that **Request-URI**.

POST

This method requests that the origin server accept the entity enclosed in the request as a new subordinate of the resource identified by the **Request-URI**.

The actual action performed by the **POST** method is determined by the origin server and usually is **Request-URI** dependent.

Responses to **POST** method are not cacheable, unless the response includes appropriate **Cache-Control** or **Expires** header fields.

PUT

This method requests that the enclosed entity be stored under the supplied **Request-URI**. If another entity exists under specified **Request-URI**, the enclosed entity should be considered as updated (newer) version of that residing on the origin server. If the **Request-URI** is not pointing to an existing resource, the origin server should create a resource with that URI.

If the request passes through a cache and the **Request-URI** identifies one or more currently cached entities, those entries should be treated as stale. Responses to this method are not cacheable.

TRACE

This method invokes a remote, application-layer loop-back of the request message. The final recipient of the request should reflect the message received back to the client as the entity-body of a 200 (OK) response. The final recipient is either the origin server or the first proxy or gateway to receive a **Max-Forwards** value of **0** in the request. A **TRACE** request must not include an entity.

Responses to this method MUST NOT be cached.

Manual:Interface/Wireless

Wireless interface configuration

Basic settings

- **master-interface** (Text) : Name of wireless interface that has *virtual-ap* capability. Virtual AP interface will only work if master interface is in *ap-bridge*, *bridge* or *wds-slave* mode. This property is only for virtual AP interfaces.
- **mode** (One of *station*, *station-wds*, *ap-bridge*, *bridge*, *alignment-only*, *nstreme-dual-slave, wds-slave*, *station-pseudobridge* or *station-pseudobridge-clone*; default value: *station*) :
    - Station modes:
        - *station* - Basic station mode. Find and connect to acceptable AP.
        - *station-wds* - Same as *station*, but create WDS link with AP, using proprietary extension. AP configuration has to allow WDS links with this device. Note that this mode does not use entries in wds.
        - *station-pseudobridge* - Same as *station*, but additionally perform MAC address translation of all traffic. Allows interface to be bridged.
        - *station-pseudobridge-clone* - Same as *station-pseudobridge*, but use station-bridge-clone-mac address to connect to AP.
    - AP modes:
        - *ap-bridge* - Basic access point mode.
        - *bridge* - Same as *ap-bridge*, but limited to one associated client.
        - *wds-slave* - Same as *ap-bridge*, but scan for AP with the same ssid and establishes WDS link. If this link is lost or cannot be established, then continue scanning. If dfs-mode is *radar-detect*, then APs with enabled hide-ssid will not be found during scanning.
    - Special modes:
        - *alignment-only* - Put interface in a continuous transmit mode that is used for aiming remote antenna.
        - *nstreme-dual-slave* - allow this interface to be used in TDMA Like Protocol “nstreme-dual” setup.

  MAC address translation in **pseudobridge** modes works by inspecting packets and building table of corresponding IP and MAC addresses. All packets are sent to AP with the MAC address used by pseudobridge, and MAC addresses of received packets are restored from the address translation table. There is single entry in address translation table for all non-IP packets, hence more than one host in the bridged network cannot reliably use non-IP protocols. Note: Currently IPv6 doesn't work over Pseudobridge
  Virtual AP interfaces do not have this property, they follow the mode of their master interface.

- **ssid** (Text, up to 32 characters long; default value is value of system/identity) : SSID (service set identifier) is a name that identifies wireless network.
- **frequency** (Frequency value in MHz) : Channel frequency on which AP will operate.

  Allowed values depend on selected band, and are restricted by country setting and wireless card capabilities.
  This setting has no effect if interface is in any of station modes, or in *wds-slave* mode, or if DFS is active.

- **band** (One of *2.4ghz-b*, *5ghz*, *5ghz-turbo*, *2.4ghz-b/g*, *2.4ghz-g-turbo*, *5ghz-10mhz*, *5ghz-5mhz*, *2ghz-10mhz*, *2ghz-5mhz*, *5ghz-11n*, *2ghz-11n*, *2.4ghz-onlyg*)
- **scan-list** (Comma separated list of frequencies and frequency ranges, or *default*) : The *default* value is all channels from selected band that are supported by card and allowed by the country and frequency-mode settings (this list can be seen in info). For default scan list in *5ghz* band channels are taken with 20MHz step, in *5ghz-turbo* band - with 40MHz step, for all other bands - with 5MHz step. If scan-list is specified manually, then all matching channels are taken. (Example: scan-list=*default,5200-5245,2412-2427* - This will use the default value of scan list for current band, and add to it supported frequencies from 5200-5245 or 2412-2427 range.)
- **antenna-mode** (One of *ant-a*, *ant-b*, *txa-rxb* or *rxa-txb*) :
  - *ant-a* - use only 'a' antenna
  - *ant-b* - use only 'b' antenna
  - *txa-rxb* - use antenna 'a' for transmitting, antenna 'b' for receiving
  - *rxa-txb* - use antenna 'b' for transmitting, antenna 'a' for receiving
- **wds-mode** (One of *disabled*, *static*, *dynamic*, *static-mesh* or *dynamic-mesh*) : Controls how WDS links with other devices (APs and clients in *station-wds* mode) are established.
  - *disabled* does not allow WDS links.
  - *static* only allows WDS links that are manually configured in wds
  - *dynamic* also allows WDS links with devices that are not configured in wds, by creating required entries dynamically. Such dynamic WDS entries are removed automatically after the connection with the other AP is lost.

  -*mesh* modes use different (better) method for establishing link between AP, that is not compatible with APs in non-mesh mode. This method avoids one-sided WDS links that are created only by one of the two APs. Such links cannot pass any data.
  When AP or station is establishing WDS connection with another AP, it uses connect-list to check whether this connection is allowed. If station in *station-wds* mode is establishing connection with AP, AP uses access-list to check whether this connection is allowed.
  If mode is *station-wds*, then this property has no effect.

- **wds-default-bridge** (*none*, or name of bridge interface) : When WDS link is established and status of the wds interface becomes *running*, it will be added as a bridge port to the bridge interface specified by this property. When WDS link is lost, wds interface is removed from the bridge. If wds interface is already included in a bridge setup when WDS link becomes active, it will not be added to bridge specified by , and will **(needs editing)**
- **wds-ignore-ssid** (*yes* or *no*; default value: *no*) : By default, WDS link between two APs can be created only when they work on the same frequency and have the same SSID value. If this property is set to *yes*, then SSID of the remote AP will not be checked. This property has no effect on connections from clients in *station-wds* mode. It also does not work if wds-mode is *static-mesh* or *dynamic-mesh*.
- **default-authentication** (*yes* or *no*, default value: *yes*) : For AP mode, this is the value of authentication for clients that do not match any entry in the access-list. For station mode, this is the value of connect for APs that do not match any entry in the connect-list.
- **default-forwarding** (*yes* or *no*; default value: *yes*) : This is the value of forwarding for clients that do not match any entry in the access-list.
- **default-ap-tx-limit** : This is the value of ap-tx-limit for clients that do not match any entry in the access-list.
- **default-client-tx-limit** : This is the value of client-tx-limit for clients that do not match any entry in the access-list.
- **hide-ssid** (*yes* or *no*; default value: *no*) :
    - *yes* - AP does not include SSID the beacon frames, and does not reply to probe requests that have broadcast SSID.
    - *no* - AP includes SSID in the beacon frames, and replies to probe requests that have broadcast SSID.

  This property has effect only in AP mode. Setting it to *yes* can remove this network from the list of wireless networks that are shown by some client software. Changing this setting does not improve security of the wireless network, because SSID is included in other frames sent by the AP.

- **security-profile** : Name of profile from security-profiles
- **compression** (*yes* or *no*; default value: *no*) : Setting this property to *yes* will allow use of the hardware compression. Wireless interface must have support for hardware compression. Connections with devices that do not use compression will still work.
- **interface-type** (*virtual-AP*, *Prism*, or *Atheros XXXXXX*, where *XXXXXX* is the type of Atheros device; read-only property) : This specifies type of wireless interface. Some properties have meaning only for certain types of interfaces.

Advanced settings

- **frequency-mode** (one of *regulatory-domain*, *manual-txpower* or *superchannel*) :

    - *regulatory-domain* - Limit available channels and maximum transmit power for each channel according to the value of country
    - *manual-txpower* - Same as above, but do not limit maximum transmit power.
    - *superchannel* - Conformance Testing Mode. Allow all channels supported by the card.

  List of available channels for each band can be seen in **/wireless info print**. This mode allows you to test wireless channels outside the default scan-list and/or regulatory domain. This mode should only be used in controlled environments, or if you have a special permission to use it in your region. Before v4.3 this was called Custom Frequency Upgrade, or Superchannel. Since HauteRouterOS v4.3 this mode is available without special key upgrades to all installations.

- **country** (either *no_country_set*, or name of a regulatory domain) : Limits available bands, frequencies and maximum transmit power for each frequency. Also specifies default value of scan-list. Value *no_country_set* is an FCC compliant set of channels.
- **antenna-gain** (default value: *0*) : Antenna gain in dBi, used to calculate maximum transmit power according to country limitations.

Default basic and supported rates, depending on selected band

band	basic rates	supported rates
2.4ghz-b	1	1 .. 11
5ghz	6	6 .. 54
2.4ghz-onlyg	6	1 .. 11 and 6 .. 54
2.4ghz-b/g	1 .. 11	1 .. 11 and 6 .. 54
2.4ghz-g-turbo	6	6 .. 54

- **rate-set** (*default* or *configured*) :
    - *default* - default basic and supported rate sets are used. Values from **basic-rates** and **supported-rates** parameters have no effect.
    - *configured* - use values from **basic-rates** and **supported-rates** parameters. Note that g mode bands use rates from both "**rates-b**" and "**rates-a/g**" properties.
- **supported-rates-b** : List of supported rates, used for *2.4ghz-b*, *2.4ghz-b/g* and *2.4ghz-onlyg* bands. Two devices will communicate only using rates that are supported by both devices. This property has effect only when value of rate-set is *configured*.
- supported-rates-a/g : Similar to the above property, but used for 5ghz, 5ghz-10mhz, 5ghz-5mhz, 5ghz-turbo, 2.4ghz-b/g, 2.4ghz-onlyg, 2ghz-10mhz, 2ghz-5mhz and 2.4ghz-g-turbo bands.

- basic-rates-b : List of basic rates, used for 2.4ghz-b, 2.4ghz-b/g and 2.4ghz-onlyg bands.

  Client will connect to AP only if it supports all basic rates announced by the AP. AP will establish WDS link only if it supports all basic rates of the other AP. This property has effect only in AP modes, and when value of rate-set is *configured*.

- **basic-rates-a/g** : Similar to the above property, but used for 5ghz, 5ghz-10mhz, 5ghz-5mhz, 5ghz-turbo, 2.4ghz-b/g, 2.4ghz-onlyg, 2ghz-10mhz, 2ghz-5mhz and 2.4ghz-g-turbo bands.
- **max-station-count** (1..2007; default value: 2007) : Maximum number of associated clients. WDS links also count toward this limit.
- **ack-timeout** (*indoors*, *dynamic*, or value in microseconds) : How long to wait for confirmation of unicast frames before considering transmission unsuccessful. Value 'dynamic' causes AP to detect and use smallest timeout that works with all connected clients.

  Acknowledgements are not used in TDMA Like “Nstreme” protocol.

- **tx-power** (value in the -30..30 range)
- **tx-power-mode** (one of *default*, *all-rates-fixed*, *card-rates* or *manual-table*) :
- **dfs-mode** (one of *none*, *no-radar-detect* and *radar-detect*; default value: *none*) : Controls DFS (Dynamic Frequency Selection).
  - *none* - disables DFS.
  - *no-radar-detect* - Select channel from scan-list with the lowest number of detected networks. In 'wds-slave' mode this setting has no effect.
  - *radar-detect* - Select channel with the lowest number of detected networks and use it if no radar is detected on it for 60 seconds. Otherwise, select different channel. This setting may be required by the country regulations.

  This property has effect only in AP mode.

- **wds-default-cost** (number; default value: *100*) : Initial bridge port cost of the WDS links.
- **wds-cost-range** (numeric range; default value: *50-150*) : Bridge port cost of WDS links are automatically adjusted, depending on measured link throughput. Port cost is recalculated and adjusted every 5 seconds if it has changed by more than 10%, or if more than 20 seconds have passed since the last adjustment.

  Setting this property to *0* disables automatic cost adjustment.
  Automatic adjustment does not work for WDS links that are manually configured as a bridge port.

- **wmm-support** (*disabled*, *enabled* or *required*) : Specifies whether to enable WMM.
- **disconnect-timeout** (time interval in the 0..15s range, in units of 10ms; default value: *3s*) : This interval is measured from third sending failure on the lowest data rate. At this point 3 * (hw-retries + 1) frame transmits on the lowest data rate had failed.

  During disconnect-timeout packet transmission will be retried with on-fail-retry-time interval. If no frame can be transmitted successfully during disconnect-timeout, connection is closed, and this event is logged as "extensive data loss". Successful frame transmission resets this timer.

- **on-fail-retry-time** (time interval in the 10ms..1s range, in 10ms units; default value: *100ms*) : After third sending failure on the lowest data rate, wait for this long before retrying.
- **frame-lifetime** (time in hundredths of a second; default value: *0*) : Discard frames that have been queued for sending longer than frame-lifetime. By default, when value of this property is *0*, frames are discarded only after connection is closed.
- **preamble-mode** (one of *long*, *short* or *both*; default value: *both*) : Short preamble mode is an option of 802.11b standard that reduces per-frame overhead.
  - On AP:
    - *long* - Do not use short preamble.
    - *short* - Announce short preamble capability. Do not accept connections from clients that do not have this capability.
    - *both* - Announce short preamble capability.
  - On station:
    - *long* - do not use short preamble.
    - *short* - do not connect to AP if it does not support short preamble.
    - *both* - Use short preamble if AP supports it.
- **allow-sharedkey** (*yes* or *no*; default value: *no*) : Allow WEP Shared Key cilents to connect. Note that no authentication is done for these clients (WEP Shared keys are not compared to anything) - they are just accepted at once (if access list allows that).
- **station-bridge-clone-mac** (MAC address) : This property has effect only in the *station-pseudobridge-clone* mode.

  Use this MAC address when connection to AP. If this value is *00:00:00:00:00:00*, station will initially use MAC address of the wireless interface.
  As soon as packet with MAC address of another device needs to be transmitted, station will reconnect to AP using that address.

- **hw-retries** (number 0..15; default value: *15*) : Number of times sending frame is retried without considering it a transmission failure.

Data rate is decreased upon failure and frame is sent again. Three sequential failures on lowest supported rate suspend transmission to this destination for the duration of on-fail-retry-time. After that, frame is sent again. The frame is being retransmitted until transmission success, or until client is disconnected after disconnect-timeout. Frame can be discarded during this time if frame-lifetime is exceeded.

- **hw-fragmentation-threshold** (integer 256..3000 | disabled;) : Specifies maximum fragment size in bytes when transmitted over wireless medium. 802.11 standard packet (MSDU in 802.11 terminology) fragmentation allows packets to be fragmented before transmitting over wireless medium to increase probability of successful transmission (only fragments that did not transmit correctly are retransmitted). Note that transmission of fragmented packet is less efficient than transmitting un-fragmented packet because of protocol overhead and increased resource usage at both - transmitting and receiving party.

Frame protection support (RTS/CTS)

802.11 standard provides means to protect transmission against other device transmission by using RTS/CTS protocol. Frame protection helps to fight "hidden node" problem. There are several types of protection:

- RTS/CTS based protection - device willing to send frame at first sends RequestToSend frame and waits for ClearToSend frame from intended destination. By "seeing" RTS or CTS frame 802.11 compliant devices know that somebody is about to transmit and therefore do not initiate transmission themselves
- "CTS to self" based protection - device willing to send frame sends CTS frame "to itself". As in RTS/CTS protocol every 802.11 compliant device receiving this frame know not to transmit. "CTS to self" based protection has less overhead, but it must be taken into account that this only protects against devices receiving CTS frame (e.g. if there are 2 "hidden" stations, there is no use for them to use "CTS to self" protection, because they will not be able to receive CTS sent by other station - in this case stations must use RTS/CTS so that other station knows not to transmit by seeing CTS transmitted by AP).

Protection mode is controlled by **hw-protection-mode** setting of wireless interface. Possible values: **none** - for no protection (default), **rts-cts** for RTS/CTS based protection or **cts-to-self** for "CTS to self" based protection.

Frame size threshold at which protection should be used is controlled by **hw-protection-threshold** setting of wireless interface.

For example, to enable "CTS-to-self" based frame protection on AP for all frames, not depending on size, use command:

```
[admin@HauteSpot] /interface wireless> set 0 hw-protection-mode=cts-to-
self hw-protection-threshold=0
```

To enable RTS/CTS based protection on client use command:

```
[admin@HauteSpot] /interface wireless> set 0 hw-protection-mode=rts-cts
hw-protection-threshold=0
```

Proprietary extensions

- **radio-name** (text) : Descriptive name of the device, that is shown in registration table entries on the remote devices.

  This is a proprietary extension.

- **area** (text; default value is empty) : Identifies group of wireless networks. This value is announced by AP, and can be matched in connect-list by area-prefix.

  This is a proprietary extension.

- **update-stats-interval** (*disabled* or time interval in the 10s..5h range; default value: *disabled*) : How often to request update of signals strength and ccq values from clients.

  Access to registration-table also triggers update of these values.
  This is proprietary extension.

- **proprietary-extensions** (*pre-2.9.25* or *post-2.9.25*; default value: *post-2.9.25*) : HauteRouterOS includes proprietary information in an information element of management frames. This parameter controls how this information is included.
  - *pre-2.9.25* - This is older method. It can interoperate with newer versions of HauteRouterOS. This method is incompatible with some clients, for example, Centrino based ones.
  - *post-2.9.25* - This uses standardized way of including vendor specific information, that is compatible with newer wireless clients.

Atheros specific

- **noise-floor-threshold** (*default* or value in the -128..127 range) : This property is only effective for cards based on AR5211 chipset.
- **adaptive-noise-immunity** (*yes* or *no*; default value: *yes*) : This property is only effective for cards based on Atheros chipset.

- **periodic-calibration** (one of *default*, *enabled* or *disabled*) : Setting *default* enables periodic calibration if info default-periodic-calibration property is *enabled*. Value of that property depends on the type of wireless card.

  This property is only effective for cards based on Atheros chipset.

- **periodic-calibration-interval** (value in range 1..10000; default value: *60*) : This property is only effective for cards based on Atheros chipset.

Prism specific

- **prism-cardtype** (*30mW*, *100mW*, *200mW*) : Specify type of the installed wireless card.

802.11n specific

- **ht-ampdu-priorities** - frame priorities for which AMPDU sending (aggregating frames and sending using block acknowledgement) should get negotiated and used. Using AMPDUs will increase throughput, but may increase latency therefore may not be desirable for real-time traffic (voice, video). Due to this, by default AMPDUs are enabled only for best-effort traffic.
- **ht-amsdu-limit** - max AMSDU that device is allowed to prepare when negotiated. AMSDU aggregation may significantly increase throughput especially for small frames, but may increase latency in case of packet loss due to retransmission of aggregated frame. Sending and receiving AMSDUs will also increase CPU usage.
- **ht-amsdu-threshold** - max frame size to allow including in AMSDU.
- **ht-basic-mcs** - Modulation and Coding Schemes that every connecting client must support (refer to 802.11n for MCS specification).
- **ht-supported-mcs** - Modulation and Coding Schemes that this device advertises as supported.
- **ht-extension-channel** - wether to use additional 20MHz extension channel and if it should be located below or above control (main) channel. Extension channel allows 11n device to use 40MHz of spectrum in total thus increasing max throughput.
- **ht-guard-interval** - whether to allow use of short guard interval (refer to 802.11n MCS specification to see how this may affect throughput). "any" will use either short or long, depending on data rate, "long" will use long.
- **ht-rxchains** - which antennas to use for receive.
- **ht-txchains** - which antetnnas to use for transmit.

Access lists

Access list is used by access point to restrict allowed connections from other devices, and to control connection parameters.

Operation

- Access list rules are checked sequentially.
- Disabled rules are always ignored.
- Only the first matching rule is applied.
- If there are no matching rules for the remote connection, then the default values from the wireless interface configuration are used.
- If remote device is matched by rule that has authentication=*no* value, the connection from that remote device is rejected.

Configuration

Access list configuration is located in **/interface wireless access-list** console path, and "Access List" tab in the HSConfig "Wireless" window.

Match properties

- **mac-address** (default value: *00:00:00:00:00:00*) : Rule matches client with the specified MAC address. Value *00:00:00:00:00:00* matches always.
- **interface** (Name of wireless interface, or *all*; default value: *all*) : Rules with interface=*all* are used for all wireless interfaces. To make rule that applies only to one wireless interface, specify that interface as a value of this property.

Match properties that also set connection parameters

- **signal-range** (*NUM..NUM* - both *NUM* are numbers in the range -120..120; default value: *-120..120*) : Rule matches if signal strength of the station is within the range.

  If signal strength of the station will go out of the range that is specified in the rule, access point will disconnect that station.

- **time** (*TIME-TIME,sun,mon,tue,wed,thu,fri,sat* - *TIME* is time interval 0..86400 seconds; all day names are optional; value can be unset; default value is not set) : Rule will match only during specified time.

  Station will be disconnected after specified time ends.
  Both start and end time is expressed as time since midnight, 00:00.
  Rule will match only during specified days of the week.

Connection properties

- **authentication** (*yes* or *no*) :
  - *no* - Client association will always fail.
  - *yes* - Use authentication procedure that is specified in the security-profile of the interface.
- **forwarding** (*yes* or *no*) :
  - *no* - Client cannot send frames to other station that are connected to same access point.
  - *yes* - Client can send frames to other stations on the same access point.
- **ap-tx-limit** (number, in bits per second; default value: *0*) : Limit rate of data transmission to this client. Value *0* means no limit.
- **client-tx-limit** (number, in bits per second; default value: *0*) : Ask client to limit rate of data transmission. Value *0* means no limit.

  This is a proprietary extension that is supported by HauteRouterOS clients.

Security related connection properties

- **private-algo** (*none*, *40bit-wep*, *104bit-wep*, *aes-ccm* or *tkip*) : Only for WEP modes.
- **private-key** : Only for WEP modes.
- **private-pre-shared-key** : Used in WPA PSK mode.

Connect lists

connect-list is used to assign priority and security settings to connections with remote access points, and to restrict allowed connections. connect-list is an ordered list of rules. Each rule in connect-list is attached to specific wireless interface, specified in the `interface` property of that rule (this is unlike access-list, where rules can apply to all interfaces). Rule can match MAC address of remote access point, it's signal strength and many other parameters.

Operation

- connect-list rules are always checked sequentially, starting from the first.
- disabled rules are always ignored.
- Only the first matching rule is applied.
- If connect-list does not have any rule that matches remote access point, then the default values from the wireless interface configuration are used.
- If access point is matched by rule that has connect=*no* value, connection with this access point will not be attempted.
- If access point is matched by rule that has connect=*yes* value, connection with this access point will be attempted.

    - In station mode, if several remote access points are matched by connect list rules with connect=*yes* value, connection will be attempted with access point that is matched by rule higher in the connect-list.
    - If no remote access points are matched by connect-list rules with connect=*yes* value, then value of default-authentication interface property determines whether station will attempt to connect to any access point. If default-authentication=*yes*, station will choose access point with best signal and compatible security.
- In access point mode, connect-list is checked before establishing WDS link with remote device. If access point is not matched by any rule in the connect list, then the value of default-authentication determines whether WDS link will be established.

Usage

Restrict station connections only to specific access points

- Set value of default-authentication interface property to *no*.

```
/interface wireless set station-wlan default-authentication=no
```

- Create rules that matches allowed access points. These rules must have connect=*yes* and interface equal to the name of station wireless interface.

```
/interface wireless connect-list add interface=station-wlan connect=yes
mac-address=00:11:22:33:00:01 /interface wireless connect-list add
interface=station-wlan connect=yes mac-address=00:11:22:33:00:02
```

Disallow connections to specific access points

- Set value of default-authentication interface property to *yes*.

```
/interface wireless set station-wlan default-authentication=yes
```

- Create connect=*no* rules that match those access points that station should not connect to. These rules must have connect=*no* and interface equal to the name of station wireless interface.

```
/interface wireless connect-list add interface=station-wlan connect=no
mac-address=00:11:22:33:44:55
```

Select preferred access points

- Create rules that match preferred access points. These rules must have connect=*yes* and interface equal to the name of station wireless interface.

- Put rules that match preferred access points higher in the connect-list, in the order of preference.

Restrict WDS link establishment

- Place rules that match allowed access points at the top.
- Add deny-all rule at the end of connect list.

Configuration Reference

Connect lists are configured under the **/interface wireless connect-list** path in the console, or in the "Connect List" tab of the "Wireless" window in the HSConfig.

Match properties

- **interface** (name of wireless interface; required) : Each rule in connect list applies only to one wireless interface that is specified by this setting.
- **area-prefix** (text) : Rule matches if area value of AP (a proprietary extension) begins with value of area-prefix.

area value is a proprietary extension.

- **mac-address** (default value: *00:00:00:00:00:00*) : Rule matches only AP with the specified MAC address. Value *00:00:00:00:00:00* matches always.
- **ssid** (text) : Rule matches access points that have this SSID. Empty value matches any SSID.

  This property has effect only when station mode interface ssid is empty, or when access point mode interface has wds-ignore-ssid=yes

Match properties that also set connection parameters

- **signal-range** (*NUM..NUM* - both *NUM* are numbers in the range -120..120) : Rule matches if signal strength of the access point is within the range.

  If station establishes connection to access point that is matched by this rule, it will disconnect from that access point when signal strength goes out of the specified range.

- **security-profile** (name of security-profile, or *none*) : Name of security profile that is used when connecting to matching access points, If value of this property is *none*, then security profile specified in the interface configuration will be used.

In station mode, rule will match only access points that can support specified security profile. Value *none* will match access point that support security profile that is specified in the interface configuration. In access point mode value of this property will not be used to match remote devices.

Connection properties

- **connect** (*yes* or *no*) :
  - *yes* - Connect to access point that matches this rule.
  - *no* - Do not connect to any access point that matches this rule.

Security profiles

Security profiles are configured under the **/interface wireless security-profiles** path in the console, or in the "Security Profiles" tab of the "Wireless" window in the HSConfig. Security profiles are referenced by the wireless interface security-profile parameter and security-profile parameter of the connect lists.

Basic properties

- **mode** (one of *none*, *static-keys-optional*, *static-keys-required* or *dynamic-keys*; default value: *none*) :
  - *none* - Encryption is not used. Encrypted frames are not accepted.
  - *static-keys-required* - WEP mode. Do not accept and do not send unencrypted frames.
    Station in *static-keys-required* mode will not connect to an access point in *static-keys-optional* mode.
  - *static-keys-optional* - WEP mode. Support encryption and decryption, but allow also to receive and send unencrypted frames. Device will send unencrypted frames if encryption algorithm is specified as *none*.
    Station in *static-keys-optional* mode will not connect to an access point in *static-keys-required* mode.
    See also: static-sta-private-algo, static-transmit-key
  - *dynamic-keys* - WPA mode.
- **name** : see generic properties

WPA properties

These properties have effect only when **mode**=*dynamic-keys*.

- **authentication-types** (multiple choice of *wpa-psk*, *wpa2-psk*, *wpa-eap* and *wpa2-eap*; default value is empty) : Set of supported authentication types. Access point will advertise supported authentication types, and client will connect to access point only if supports any of the advertised authentication types.

- **unicast-ciphers** (multiple choice of *tkip*, *aes-ccm*; default value is empty) : Access point advertises that it supports specified ciphers. Client attempts connection only to access points that supports at least one of the specified ciphers.

  One of the ciphers will be used to encrypt unicast frames that are sent between access point and station.

- **group-ciphers** (multiple choice of *tkip*, *aes-ccm*; default value is empty) : Access point advertises one of these ciphers, and uses it to encrypt all broadcast and multicast frames. Client attempts connection only to access points that use one of the specified group ciphers.
- **group-key-update** (time interval in the 30s..1h range; default value: *5m*) : Controls how often access point updates group key. This key is used to encrypt all broadcast and multicast frames.

  This property has no effect in station mode.

- **wpa-pre-shared-key**, **wpa2-pre-shared-key** (text) : WPA and WPA2 pre-shared key mode requires all devices in a BSS to have common secret key. Value of this key can be an arbitrary text.

HauteRouterOS also allows to override pre-shared key value for specific clients, using either **private-pre-shared-key** property in the access-list, or the HauteSpot-Wireless-Psk attribute in the RADIUS MAC authentication response. This is an extension.

These properties have effect only when **authentication-types** contains either *wpa-psk* or *wpa2-psk*.

wpa-pre-shared-key is used for *wpa-psk* authentication type. **wpa2-pre-shared-key** is used for *wpa2-psk*.

WPA EAP properties

These properties have effect only when **authentication-types** contains *wpa-eap* or *wpa2-eap*, and **mode**=*dynamic-keys*.

- **eap-methods** (array of *eap-tls*, *passthrough*) :
  - *eap-tls* - Use built-in EAP TLS authentication. Both client and server certificates are supported. See description of **tls-mode** and **tls-certificate** properties.
  - *passthrough* - Access point will relay authentication process to the RADIUS server. This value is ignored in station mode.

Order of values is significant for access point configuration, it is used by access point when offering specified methods to clients.
Example: Access point uses security-profile where **eap-methods**=*eap-tls*,*passthrough*:
Access point offers EAP-TLS method to the client.
Client refuses.
Access point starts relaying EAP communication to the radius server.

- **supplicant-identity** (text; default value is same as system/identity of router at the moment of profile creation) : EAP identity that is sent by client at the beginning of EAP authentication. This value is used as a value for User-Name attribute in RADIUS messages sent by RADIUS EAP accounting and RADIUS EAP pass-through authentication.
- **tls-mode** (one of *verify-certificate*, *dont-verify-certificate*, *no-certificates*; default value: *no-certificates*) :
  - *verify-certificate* - Require remote device to have valid certificate. Check that it is signed by known certificate authority. No additional identity verification is done.
    Note: Certificate may include information about time period during which it is valid. If router has incorrect time and date, it may reject valid certificate because router's clock is outside that period.
    See also: certificate configuration.
  - *dont-verify-certificate* - Do not check certificate of the remote device. Access point will not require client to provide certificate.
  - *no-certificates* - Do not use certificates. TLS session is established using 2048 bit anonymous Diffie-Hellman key exchange.

  When using first two modes, remote device has to support one of the "RC4-MD5", "RC4-SHA" or "DES-CBC3-SHA" TLS cipher suites. In the last mode remote device must support "ADH-DES-CBC3-SHA" cipher suite.
  This property has effect only when **eap-methods** contains *eap-tls*.

- **tls-certificate** (*none* or name of certificate; default value: *none*) : Access point always needs certificate when configured with **tls-mode**=*verify-certificate*, or **tls-mode**=*dont-verify-certificate*. Client needs certificate only if *access point* is configured with **tls-mode**=*verify-certificate*. In this case client needs valid certificate that is signed by CA known to the access point.

  This property has effect only if **tls-mode**≠*no-certificates*.
  This property has effect only when **eap-methods** contains *eap-tls*.

RADIUS properties

- **radius-mac-authentication** (*yes* or *no*; default value: *no*) : This property affects the way how access point processes clients that are not found in the access-list.
  - *no* - allow or reject client authentication based on the value of **default-authentication** property of the wireless interface.
  - *yes* - Query RADIUS server using MAC address of client as user name. With this setting the value of **default-authentication** has no effect.
- **radius-mac-accounting** (*yes* or *no*; default value: *no*) : use mac address when reporting back to radius for accounting purposes
- **radius-eap-accounting** (*yes* or *no*; default value: *no*) : use EAP account information when reporting back to radius for accounting purposes
- **interim-update** (time interval; default value: *0*) : When RADIUS accounting is used, access point periodically sends accounting information updates to the RADIUS server. This property specifies default update interval that can be overridden by the RADIUS server using Acct-Interim-Interval attribute.
- **radius-mac-format** (one of *XX:XX:XX:XX:XX:XX*, *XXXX:XXXX:XXXX*, *XXXXXX:XXXXXX*, *XX-XX-XX-XX-XX-XX*, *XXXXXX-XXXXXX*, *XXXXXXXXXXXX*, *XX XX XX XX XX XX*; default value: *XX:XX:XX:XX:XX:XX*) : Controls how MAC address of the client is encoded by access point in the User-Name attribute of the MAC authentication and MAC accounting RADIUS requests.
- **radius-mac-mode** (one of *as-username*, *as-username-and-password*; default value: *as-username*) : By default access point uses empty password, when sending Access-Request during MAC authentication. When this property is set to *as-username-and-password*, access point will use the same value for User-Password attribute as for the User-Name attribute.
- **radius-mac-caching** (either *disabled* or time interval; default value: *disabled*) : If this value is set to time interval, the access point will cache RADIUS MAC authentication responses for specified time, and will not contact RADIUS server if matching cache entry already exists. Value *disabled* will disable cache, access point will always contact RADIUS server.

WEP properties

These properties have effect only when **mode** is *static-keys-required* or *static-keys-optional*. See section "Wireless#Statically_configured_WEP_keys".

- **static-key-0**, **static-key-1**, **static-key-2**, **static-key-3** (hexadecimal representation of the key. Length of key must be appropriate for selected algorithm - see section "Statically configured WEP keys; default value is empty) :
- **static-algo-0**, **static-algo-1**, **static-algo-2**, **static-algo-3** (one of *none*, *40bit-wep*, *104bit-wep*, *tkip* or *aes-ccm*; default value: *none*) : Encryption algorithm to use with the corresponding key.
- **static-transmit-key** (one of *key-0*, *key-1*, *key-2* or *key-3*; default value: *key-0*) : Access point will use the specified key to encrypt frames for clients that do not

use private key. Access point will also use this key to encrypt broadcast and multicast frames.

Client will use the specified key to encrypt frames if **static-sta-private-algo**=*none*.
If corresponding **static-algo-** property has value *none*, frame will be sent unencrypted (when **mode**=*static-keys-optional*) or will not be sent at all (when **mode**=*static-keys-required*).

- **static-sta-private-key** (hexadecimal representation of the key. Length of key must be appropriate for selected algorithm - see section "Statically configured WEP keys") : This property is used only in station mode. Access point uses corresponding key either from **private-key** property of access-list, or from HauteSpot-Wireless-Enc-Key attribute in RADIUS Access-Accept MAC authentication response.
- **static-sta-private-algo** (one of *none*, *40bit-wep*, *104bit-wep*, *tkip* or *aes-ccm*) : Encryption algorithm to use with station private key. Value *none* disables use of the private key.

  This property is used only in station mode. Access point has to get corresponding value either from **private-algo** property of access-list, or from HauteSpot-Wireless-Enc-Algo attribute in RADIUS Access-Accept MAC authentication response.
  Station private key replaces key 0 for unicast frames. Station will not use private key to decrypt broadcast frames.

Management frame protection

***Used for**: Deauthentication attack prevention, MAC address cloning issue.*

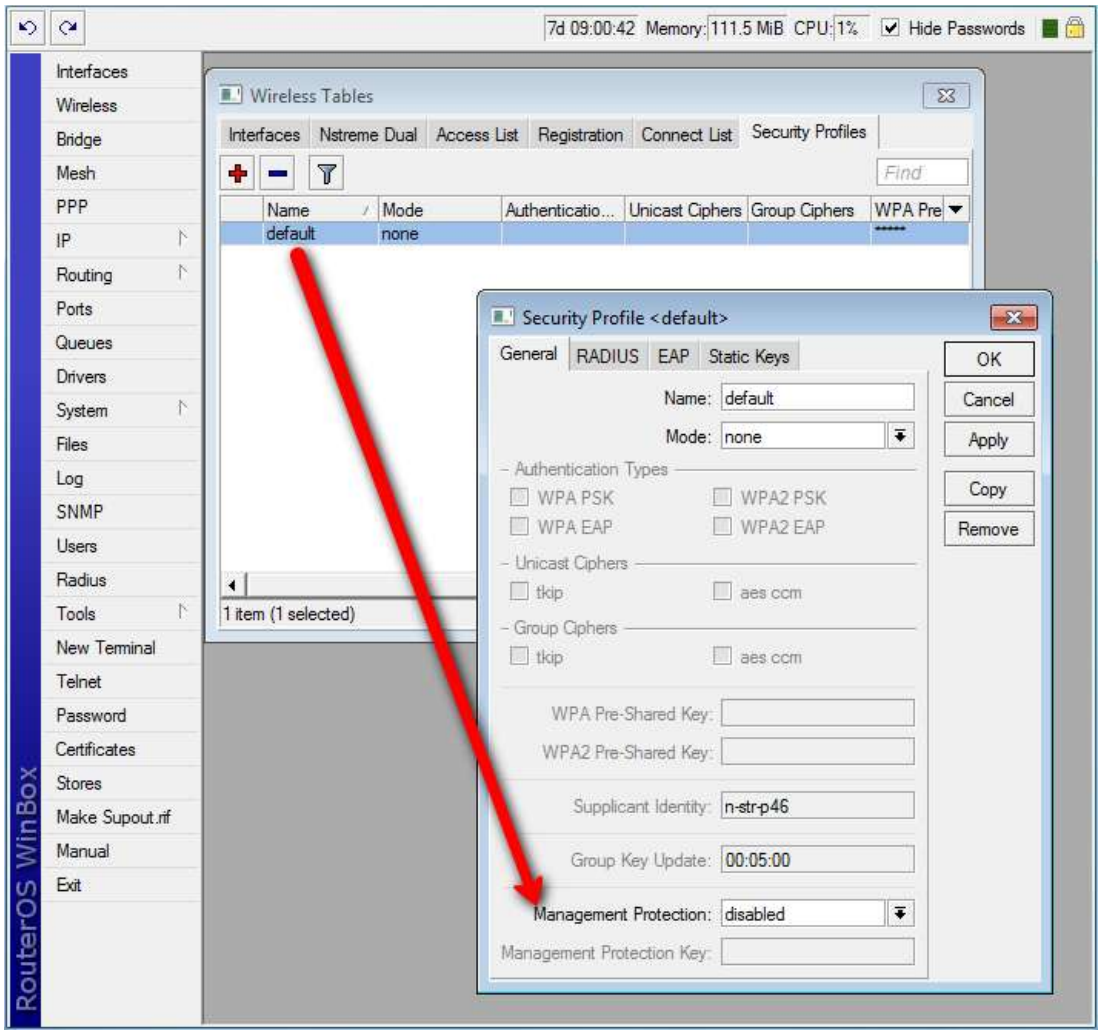
HauteRouterOS implements proprietary management frame protection algorithm based on shared secret. Management frame protection means that HauteRouterOS wireless device is able to verify source of management frame and confirm that particular frame is not malicious. This feature allows to withstand deauthentication and disassociation attacks on HauteRouterOS based wireless devices.

Management protection mode is configured in security-profile with **management-protection** setting. Possible values are: **disabled** - management protection is disabled (default), **allowed** - use management protection if supported by remote party (for AP - allow both, non-management protection and management protection clients, for client - connect both to APs with and without management protection), **required** - establish association only with remote devices that support management protection (for AP - accept only clients that support management protection, for client - connect only to APs that support management protection).

Management protection shared secret is configured with security-profile **management-protection-key** setting.

When interface is in AP mode, default management protection key (configured in security-profile) can be overridden by key specified in access-list or RADIUS attribute.

```
[admin@HauteSpot] /interface wireless security-profiles> print
 0 name="default" mode=none authentication-types="" unicast-ciphers=""
  group-ciphers="" wpa-pre-shared-key="" wpa2-pre-shared-key=""
  supplicant-identity="n-str-p46" eap-methods=passthrough
  tls-mode=no-certificates tls-certificate=none static-algo-0=none
  static-key-0="" static-algo-1=none static-key-1="" static-algo-2=none
  static-key-2="" static-algo-3=none static-key-3=""
  static-transmit-key=key-0 static-sta-private-algo=none
  static-sta-private-key="" radius-mac-authentication=no
  radius-mac-accounting=no radius-eap-accounting=no interim-update=0s
  radius-mac-format=XX:XX:XX:XX:XX:XX radius-mac-mode=as-username
  radius-mac-caching=disabled group-key-update=5m
  management-protection=disabled management-protection-key=""
[admin@HauteSpot] /interface wireless security-profiles> set default
management-protection=
                                                        allowed
disabled  required
```



Operation details

RADIUS MAC authentication

Note: RAIDUS MAC authentication is used by access point for clients that are not found in the access-list, similarly to the **default-authentication** property of the wireless interface. It controls whether client is allowed to proceed with authentication, or is rejected immediately.

When **radius-mac-authentication**=*yes*, access point queries RADIUS server by sending Access-Request with the following attributes:

- User-Name - Client MAC address. This is encoded as specified by the **radius-mac-format** setting. Default encoding is "XX:XX:XX:XX:XX:XX".

- Nas-Port-Id - name of wireless interface.
- User-Password - When radius-mac-mode=*as-username-and-password* this is set to the same value as User-Name. Otherwise this attribute is empty.
- Calling-Station-Id - Client MAC address, encoded as "XX-XX-XX-XX-XX-XX".
- Called-Station-Id - MAC address and SSID of the access point, encoded as "XX-XX-XX-XX-XX-XX:SSID" (minus separated pairs of MAC address digits, followed by colon, followed by SSID value).
- Acct-Session-Id - Added when radius-mac-accounting=*yes*.

When access point receives Access-Accept or Access-Reject response from the RADIUS server, it stores the response and either allows or rejects client. Access point uses following RADIUS attributes from the Access-Accept response:

- Ascend-Data-Rate
- Ascend-Xmit-Rate
- HauteSpot-Wireless-Forward - Same as access-list forwarding.
- HauteSpot-Wireless-Enc-Algo - Same as access-list private-algo.
- HauteSpot-Wireless-Enc-Key - Same as access-list private-key.
- HauteSpot-Wireless-Psk - Same as access-list private-pre-shared-key.
- Session-Timeout - Time, after which client will be disconnected.
- Acct-Interim-Interval - Overrides value of interim-update.
- Class - If present, value of this attribute is saved and included in Accounting-Request messages.

Caching

Caching of RADIUS MAC authentication was added to support RADIUS authentication for clients that require from the access point very quick response to the association request. Such clients time out before response from RADIUS server is received. Access point caches authentication response for some time and can immediately reply to the repeated association request from the same client.

RADIUS EAP pass-through authentication

When using WPA EAP authentication type, clients that have passed MAC authentication are required to perform EAP authentication before being authorized to pass data on wireless network. With pass-through EAP method the access point will relay authentication to RADIUS server, and use following attributes in the Access-Request RADIUS message:

- User-Name - EAP supplicant identity. This value is configured in the supplicant-identity property of the client security profile.
- Nas-Port-Id - name of wireless interface.
- Calling-Station-Id - Client MAC address, encoded as "XX-XX-XX-XX-XX-XX".

- Called-Station-Id - MAC address and SSID of the access point, encoded as "XX-XX-XX-XX-XX-XX:SSID" (pairs of MAC address digits separated by minus sign, followed by colon, followed by SSID value).
- Acct-Session-Id - Added when radius-eap-accounting=*yes*.
- Acct-Multi-Session-Id - MAC address of access point and client, and unique 8 byte value, that is shared for all accounting sessions that share single EAP authentication. Encoded as *AA-AA-AA-AA-AA-AA-CC-CC-CC-CC-CC-CC-XX-XX-XX-XX-XX-XX-XX-XX*.

  Added when radius-eap-accounting=*yes*.

Access point uses following RADIUS attributes from the Access-Accept server response:

- Class - If present, value of this attribute is saved and included in Accounting-Request messages.
- Session-Timeout - Time, after which client will be disconnected. Additionally, access point will remember authentication result, and if during this time client reconnects, it will be authorized immediately, without repeating EAP authentication.
- Acct-Interim-Interval - Overrides value of interim-update.

Statically configured WEP keys

Different algorithms require different length of keys:

- *40bit-wep* - 10 hexadecimal digits (40 bits). If key is longer, only first 40 bits are used.
- *104bit-wep* - 26 hexadecimal digits (104 bits). If key is longer, only first 104 bits are used.
- *tkip* - At least 64 hexadecimal digits (256 bits).
- *aes-ccm* - At least 32 hexadecimal digits (128 bits).

Key must contain even number of hexadecimal digits.

WDS security configuration

WDS links can use all available security features. However, they require careful configuration of security parameters.

It is possible to use one security profile for all clients, and different security profiles for WDS links. Security profile for WDS link is specified in connect-list. Access point always checks connect list before establishing WDS link with another access point, and used security settings from matching connect list entry. WDS link will work when each

access point will have connect list entry that matches the other device, has **connect**=*yes* and specifies compatible **security-profile**.

WDS and WPA/WPA2

If access point uses security profile with **mode**=*dynamic-keys*, then encryption will be used for all WDS links. Since WPA authentication and key exchange is not symmetrical, one of the access points will act as a client for the purpose of establishing secure connection. This is similar to how *static-mesh* and *dynamic-mesh* WDS modes work. Some problems, like single sided WDS link between two incorrectly configured access points that use non-*mesh* mode, is not possible if WPA encryption is enabled. However, non-*mesh* modes with WPA still have other issues (like constant reconnection attempts in case of configuration mismatch) that are solved by use of the -*mesh* WDS modes.

In general, WPA properties on both access points that establish WPA protected WDS link have to match. These properties are **authentication-types**, **unicast-ciphers**, **group-ciphers**. For non-*mesh* WDS mode these properties need to have the same values on both devices. In *mesh* WDS mode each access point has to support the other one as a client.

Theoretically it is possible to use RADIUS MAC authentication and other RADIUS services with WDS links. However, only one access point will interact with the RADIUS server, the other access point will behave as a client.

Implementation of *eap-tls* EAP method in HauteRouterOS is particularly well suited for WDS link encryption. **tls-mode**=*no-certificates* requires no additional configuration, and provides very strong encryption.

WDS and WEP

mode, static-sta-private-key and static-sta-private-algo parameters in the security profile assigned to the WDS link need to have the same values on both access points that establish WDS link with WPA encryption.

Security profile and access point matching in the connect list

Client uses value of connect-list security-profile property to match only those access points that support necessary security.

- **mode**=*static-keys-required* and mode=*static-keys-optional* matches only access points with the same mode in interface security-profile.
- If mode=*dynamic-keys*, then connect list entry matches if all of the authentication-types, unicast-ciphers and group-ciphers contain at least one value that is advertised by access point.

Wireless Debug Logs

Debugging wireless problems using Logs.

By default HauteRouterOS wireless log shows that client connects and disconnects as simple entries:

```
22:32:18 wireless,info 00:80:48:41:AF:2A@wlan1: connected
```

It is enough for regular users to know that the wireless client with MAC address "00:80:48:41:AF:2A" is connected to wireless interface "wlan1". But actually there are more log entries available than are shown in standard logging. They are called 'debug' logs which give more detailed information. In the following Debug Log example you will see the same client connecting to the AP in more detail than found in typical logging:

```
22:33:20 wireless,debug wlan1: 00:80:48:41:AF:2A attempts to connect
22:33:20 wireless,debug wlan1: 00:80:48:41:AF:2A not in local ACL, by
default accept
22:33:20 wireless,info 00:80:48:41:AF:2A@wlan1: connected
```

Debug Logs will give you more specific information on each step of the Client wireless connection and disconnection. The first line shows that the wireless client tried to connect to the AP. On the second line the AP checked to see if that client is allowed to connect to the AP and the resulting action. And only on the third line do you see that the client is connected. This is merely one example of the debug log messages. The description of all debug entries is written below.

To enable the wireless debug logs you should execute such commands:

```
[admin@HauteSpot] > /system logging
[admin@HauteSpot] system logging> add topics=wireless,debug
action=memory
```

You can also use HS Configurator instead command line. Simply click the 'System' menu and in the sub menu select 'Logging'. Add an additional logging rule using the button. You will need to create an additional topic using the down arrow on the very right of the 'Topics:' field. In Figure 1 below you can see the correct configuration of the log rule.

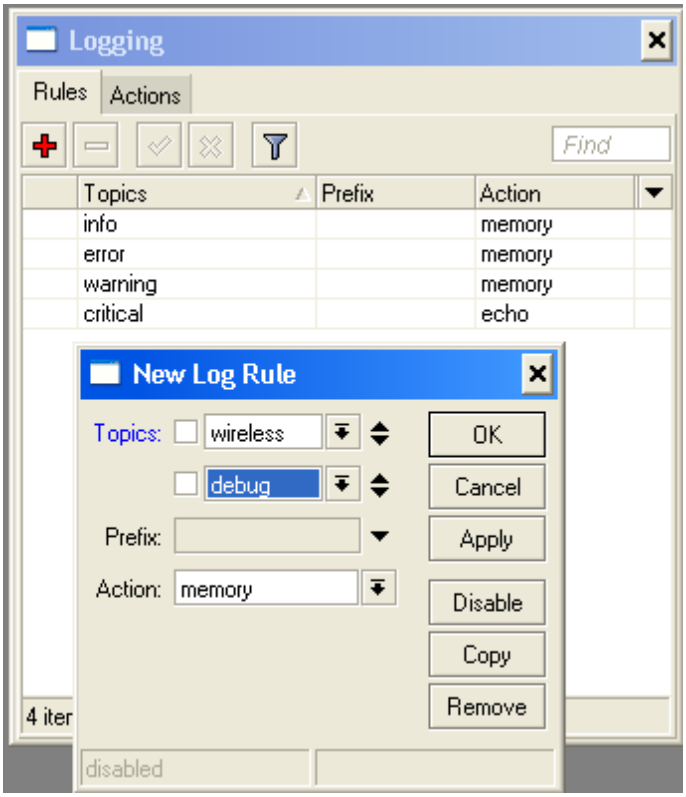


Figure 1: HSconf method

This will help you understand and fix wireless problems with ease and with less interaction with the support team.

STATION MODE

<MAC>@<DEV>: lost connection, <REASON>

Station has lost connection to AP because of <REASON>

<MAC>@<DEV>: failed to connect, <REASON>

Station attempted to connect to AP, but failed due to <REASON>

<MAC>@<DEV>: established connection on <FREQ>, SSID <SSID>

Station attempted and succesfully connected to AP with SSID <SSID> on frequency <FREQ>.

<MAC>@<DEV>: MIC failure!!!

TKIP message integrity check failure, somebody must be trying to break into or DOS network, If more than 1 MIC failure is encountered during 60s period, "TKIP countermeasures" state is entered.

<MAC>@<DEV>: enter TKIP countermeasures

Entered TKIP countermeasures state, this means that Station will disconnect from AP and keep silence for 60s.

AP MODE

<DEV>: radar detected on <FREQ>

Radar detected on frequency <FREQ>, AP will look for other channel

<DEV>: data from unknown device <MAC>, sent deauth [(XXX events suppressed, YYY deauths suppressed)]

Data frame from unknown device (read - not registered to this AP) with mac address <MAC> received, AP sent deauthentication frame to it (as per 802.11). XXX is number of events that are not logged so that the log does not become too large (logs are limited to 1 entry per 5s after first 5 entries), YYY is the number of deauthentication frames that should have been sent, but were not sent, so that resources are not wasted sending too many deauthentication frames (only 10 deauth frames per second are allowed).

The likely cause of such a message is that the Station previously connected to the AP, which does not yet know it has been dropped from AP registration table, sending data to AP. Deauthentication message tells the Station that it is no longer connected.

<DEV>: denying assoc to <MAC>, failed to setup compression

Failed to initialize compression on AP, most likely because there are too many clients attempting to connect and use compression.

<DEV>: <MAC> is new WDS master

WDS slave has established connection to WDS master, this means that WDS slave starts accepting clients and acting as AP.

<DEV>: <MAC> was WDS master

This message appears after connection with <MAC> is lost, means that WDS slave will disconnect all clients and start scanning to find new WDS master.

<MAC>@<DEV>: connected [, is AP][, wants WDS]

Station with address <MAC> connected. if "is AP" present - remote device is AP, if "is WDS" presents, remote device wants to establish WDS link.

<MAC>@<DEV>: disconnected, <REASON>

Connection with Station with address <MAC> terminated due to <REASON>

<DEV>: TKIP countermeasures over, resuming

TKIP countermeasures (60s silence period) over, AP resumes acting as AP.

<DEV>: starting TKIP countermeasures

Entering TKIP countermeasures state (60s silence period), all clients will be lost.

<REASON>

"joining failed" - can only happen on Prism cards in station mode, failed to connect to AP due to some reason

"join timeout" - happens on Station, failed to synchronize to AP (receive first beacon frame). Most likely weak signal, remote turned off, strong interference, some other RF related issue that makes communication impossible.

"no beacons" - no beacons received from remote end of WDS link. Most likely weak signal, remote turned off, strong interference, some other RF related issue that makes communication impossible.

"extensive data loss" - local interface decided to drop connection to remote device because of inability to send data to remote after multiple failures at lowest possible rate. Possible causes - too weak signal, remote device turned off, strong interference, some other RF related issue that makes communication impossible.

"decided to deauth, <802.11 reason>" - local interface decided do deauthenticate remote device using 802.11 reason <802.11 reason>.

"inactivity" - remote device was inactive for too long

"device disabled" - local interface got disabled

"got deauth, <802.11 reason>" - received deauthentication frame from remote device, 802.11 reason code is reported in <802.11 reason>

"got disassoc, <802.11 reason>" - received disassociation frame from remote device, 802.11 reason code is reported in <802.11 reason>

"auth frame from AP" - authentication frame from remote device that is known to be AP, most likely mode changes on remote device from AP to Station.

"bad ssid" - bad ssid for WDS link

"beacon from non AP" - received beacon frame from remote device that is known to be non-AP node, most likely mode changes on remote device from Station to AP.

"no WDS support" - does not report WDS support

"failed to confirm SSID" - failed to confirm SSID of other end of WDS link.

"hardware failure" - some hardware failure or unexpected behaviour. Not likely to be seen.

"lost connection" - can only happen on Prism cards in station mode, connection to AP lost due to some reason.

"auth failed <802.11 status>" - happens on Station, AP denies authentication, 802.11 status code is reported in <802.11 status>.

"assoc failed <802.11 status>" - happens on Station, AP denies association, 802.11 status code is reported in <802.11 status>.

"auth timeout" - happens on Station, Station does not receive response to authentication frames, either bad link or AP is ignoring this Station for some reason.

"assoc timeout" - happens on Station, Station does not receive response to association frames, either bad link or AP is ignoring this Station for some reason.

"reassociating" - happens on AP: connection assumed to be lost, because Station that is considered already associated attempts to associate again. All connection related information must be deleted, because during association process connection parameters are negotiated (therefore "disconnected"). The reason why Station reassociates must be looked for on Station (most likely cause is that Station for some reason dropped connection without telling AP - e.g. data loss, configuration changes).

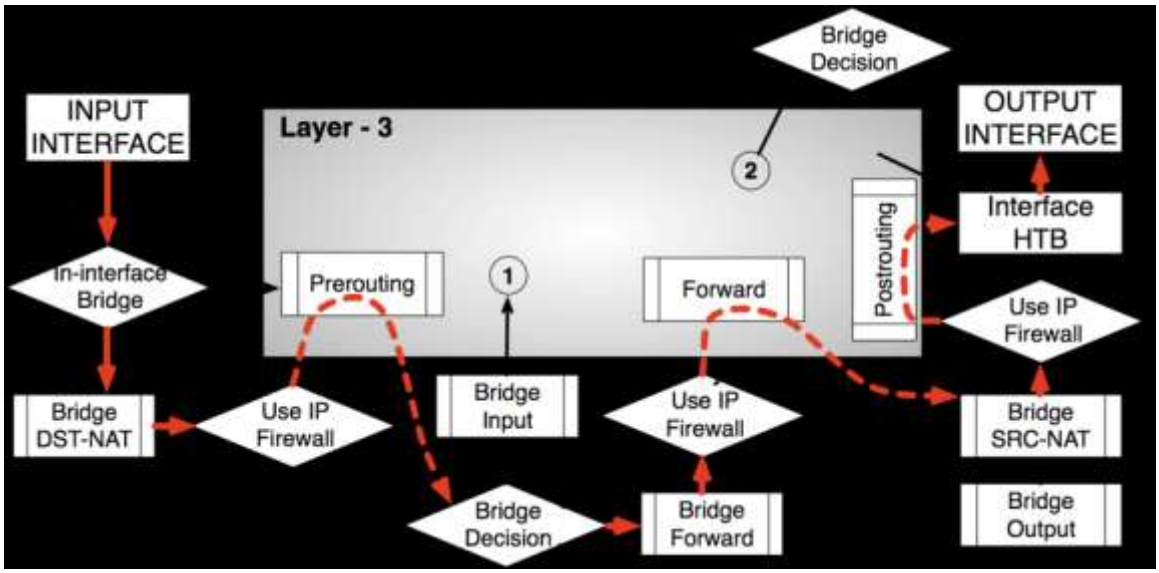
"compression setup failure" - connection impossible, because not enough resources to do compression (too many stations that want to use compression already connected)

Bridge Firewall

Sub-menu: `/interface bridge filter, /interface bridge nat`

The bridge firewall implements packet filtering and thereby provides security functions that are used to manage data flow to, from and through bridge.

Packet flow diagram shows how packets are processed through router. It is possible to force bridge traffic to go through `/ip firewall filter` rules



There are two bridge firewall tables:

- **filter** - bridge firewall with three predefined chains:

    - **input** - filters packets, which destination is the bridge (including those packets that will be routed, as they are anyway destined to the bridge MAC address)
    - **output** - filters packets, which come from the bridge (including those packets that has been routed normally)
    - **forward** - filters packets, which are to be bridged (note: this chain is not applied to the packets that should be routed through the router, just to those that are traversing between the ports of the same bridge)
- **nat** - bridge network address translation provides ways for changing source/destination MAC addresses of the packets traversing a bridge. Has two built-in chains:
    - **srcnat** - used for "hiding" a host or a network behind a different MAC address. This chain is applied to the packets leaving the router through a bridged interface
    - **dstnat** - used for redirecting some pakets to another destinations

You can put packet marks in bridge firewall (filter and NAT), which are the same as the packet marks in IP firewall put by mangle. So packet marks put by bridge firewall can be used in IP firewall, and vice versa.

General bridge firewall properties are described in this section. Some parameters that differ between nat and filter rules are described in further sections.

Properties

Property	Description
802.3-sap (*integer*)	DSAP (Destination Service Access Point) and SSAP (Source Service Access Point) are 2 one byte fields, which identify the network protocol entities which use the link layer service. These bytes are always equal. Two hexadecimal digits may be specified here to match an SAP byte
802.3-type (*integer*)	Ethernet protocol type, placed after the IEEE 802.2 frame header. Works only if 802.3-sap is 0xAA (SNAP - Sub-Network Attachment Point header). For example, AppleTalk can be indicated by SAP code of 0xAA followed by a SNAP type code of 0x809B
arp-dst-address (*IP address; default:*)	ARP destination address
arp-dst-mac-address (*MAC address; default:*)	ARP destination MAC address
arp-gratuitous (*yes* | *no; default:*)	Matches ARP gratuitous packets

arp-hardware-type (*integer; default: 1*)

ARP hardware type. This normally Ethernet (Type 1)

arp-opcode (*arp-nak* | *drarp-error* | *drarp-reply* | *drarp-request* | *inarp-reply* | *inarp-request* | *reply* | *reply-reverse* | *request* | *request-reverse*)

ARP opcode (packet type)

- **arp-nak** - negative ARP reply (rarely used, mostly in ATM networks)
- **drarp-error** - Dynamic RARP error code, saying that an IP address for the given MAC address can not be allocated
- **drarp-reply** - Dynamic RARP reply, with a temporaty IP address assignment for a host
- **drarp-request** - Dynamic RARP request to assign a temporary IP address for the given MAC address
- **inarp-reply** -
- **inarp-request** -
- **reply** - standard ARP reply with a MAC address
- **reply-reverse** - reverse ARP (RARP) reply with an IP address assigned
- **request** - standard ARP request to a known IP address to find out unknown MAC address
- **request-reverse** - reverse ARP (RARP) request to a known MAC address to find out unknown IP address (intended to be used by hosts to find out their own IP address, similarly to DHCP service)

arp-src-address (*IP address; default:*)

ARP source address

arp-src-mac-address (*MAC address; default:*)

ARP source MAC address

chain (*text*)

Bridge firewall chain, which the filter is functioning in (either a built-in one, or a user defined)

dst-address (*IP address; default:*)

Destination IP address (only if MAC protocol is set to IPv4)

dst-mac-address (*MAC address; default:*)

Destination MAC address

dst-port (*integer 0..65535*)

Destination port number or range (only for TCP or UDP protocols)

in-bridge (*name*)

Bridge interface through which the packet is coming in

in-interface (*name*)

Physical interface (i.e., bridge port) through which the packet is coming in

ingress-priority (*integer 0..63*)

Matches ingress priority of the packet. Priority may be derived from VLAN, WMM or MPLS EXP bit. read more»

ip-protocol (*ddp* | *ggp* | *icmp* | *igmp* | *ipsec-ah* | *ospf* | *rdp* | *tcp* | *vrrp* | *egp* | *gre* | *icmpv6* | *ipencap* | *ipsec-esp* | *pim* | *rspf* | *udp* | *xns-idp* | *encap* | *hmp* | *idpr-cmtp* | *ipip* | *iso-tp4* | *pup* | *st* | *vmtp* | *xtp*)

IP protocol (only if MAC protocol is set to IPv4)

- **ipsec-ah** - IPsec AH protocol
- **ipsec-esp** - IPsec ESP protocol
- **ddp** - datagram delivery protocol
- **egp** - exterior gateway protocol
- **ggp** - gateway-gateway protocol
- **gre** - general routing encapsulation
- **hmp** - host monitoring protocol
- **idpr-cmtp** - idpr control message transport
- **icmp** - internet control message protocol
- **icmpv6** -
- **igmp** - internet group management protocol
- **ipencap** - ip encapsulated in ip
- **encap** - ip encapsulation
- **ipip** - ip encapsulation
- **iso-tp4** - iso transport protocol class 4
- **ospf** - open shortest path first
- **pim** - protocol independent multicast
- **pup** - parc universal packet protocol
- **rspf** - radio shortest path first
- **rdp** - reliable datagram protocol
- **st** - st datagram mode
- **tcp** - transmission control protocol
- **udp** - user datagram protocol
- **vmtp** - versatile message transport
- **vrrp** -
- **xns-idp** - xerox ns idp
- **xtp** – xpress transfer protocol

jump-target (*name*)

If `action=jump` specified, then specifies the user-defined firewall chain to process the packet

limit (*integer/time,integer*)

Restricts packet match rate to a given limit.

- **count** - maximum average packet rate, measured in packets per second (pps), unless followed by Time option
- **time** - specifies the time interval over which

	the packet rate is measured • **burst** - number of packets to match in a burst
log-prefix (*text*)	Defines the prefix to be printed before the logging information
mac-protocol (*arp* | *ip* | *ipv6* | *ipx* | *length* | *pppoe* | *pppoe-discovery* | *rarp* | *vlan*)	Ethernet payload type (MAC-level protocol)
out-bridge (*name*)	Outgoing bridge interface
out-interface (*name*)	Interface via packet is leaving the bridge
packet-mark (*name*)	Match packets with certain packet mark
packet-type (*broadcast* | *host* | *multicast* | *other-host*)	MAC frame type: • **broadcast** - broadcast MAC packet • **host** - packet is destined to the bridge itself • **multicast** - multicast MAC packet • **other-host** - packet is destined to some other unicast address, not to the bridge itself
src-address (*IP address; default:*)	Source IP address (only if MAC protocol is set to IPv4)
src-mac-address (*MAC address; default:*)	Source MAC address
src-port (*integer 0..65535*)	Source port number or range (only for TCP or UDP protocols)
stp-flags (*topology-change* | *topology-change-ack*)	The BPDU (Bridge Protocol Data Unit) flags. Bridge exchange configuration messages named BPDU peridiocally for preventing from loop • **topology-change** - topology change flag is set when a bridge detects port state change, to force all other bridges to drop their host tables and recalculate network topology • **topology-change-ack** - topology change acknowledgement flag is sen in replies to the notification packets
stp-forward-delay (*time 0..65535*)	Forward delay timer
stp-hello-time (*time 0..65535*)	STP hello packets time
stp-max-age (*time 0..65535*)	Maximal STP message age
stp-msg-age (*time 0..65535*)	STP message age

stp-port (*integer 0..65535*)	STP port identifier
stp-root-address (*MAC address*)	Root bridge MAC address
stp-root-cost (*integer 0..65535*)	Root bridge cost
stp-root-priority (*integer 0..65535*)	Root bridge priority
stp-sender-address (*MAC address*)	STP message sender MAC address
stp-sender-priority (*integer 0..65535*)	STP sender priority
stp-type (*config* | *tcn*)	The BPDU type: • **config** - configuration BPDU • **tcn** - topology change notification
vlan-encap (*arp* | *ip* | *ipv6* | *ipx* | *length* | *pppoe* | *pppoe-discovery* | *rarp* | *vlan*)	the MAC protocol type encapsulated in the VLAN frame
vlan-id (*integer 0..4095*)	VLAN identifier field
vlan-priority (*integer 0..7*)	The user priority field

Notes

- STP matchers are only valid if destination MAC address is 01:80:C2:00:00:00/FF:FF:FF:FF:FF:FF (Bridge Group address), also `stp` should be enabled.
- ARP matchers are only valid if `mac-protocol` is `arp` or `rarp`
- VLAN matchers are only valid for `vlan` ethernet protocol
- IP-related matchers are only valid if `mac-protocol` is set as `ipv4`
- 802.3 matchers are only consulted if the actual frame is compliant with IEEE 802.2 and IEEE 802.3 standards (**note**: it is not the industry-standard Ethernet frame format used in most networks worldwide!). These matchers are ignored for other packets.

Bridge Packet Filter

Sub-menu: `/interface bridge filter`

This section describes bridge packet filter specific filtering options, which were omitted in the general firewall description.

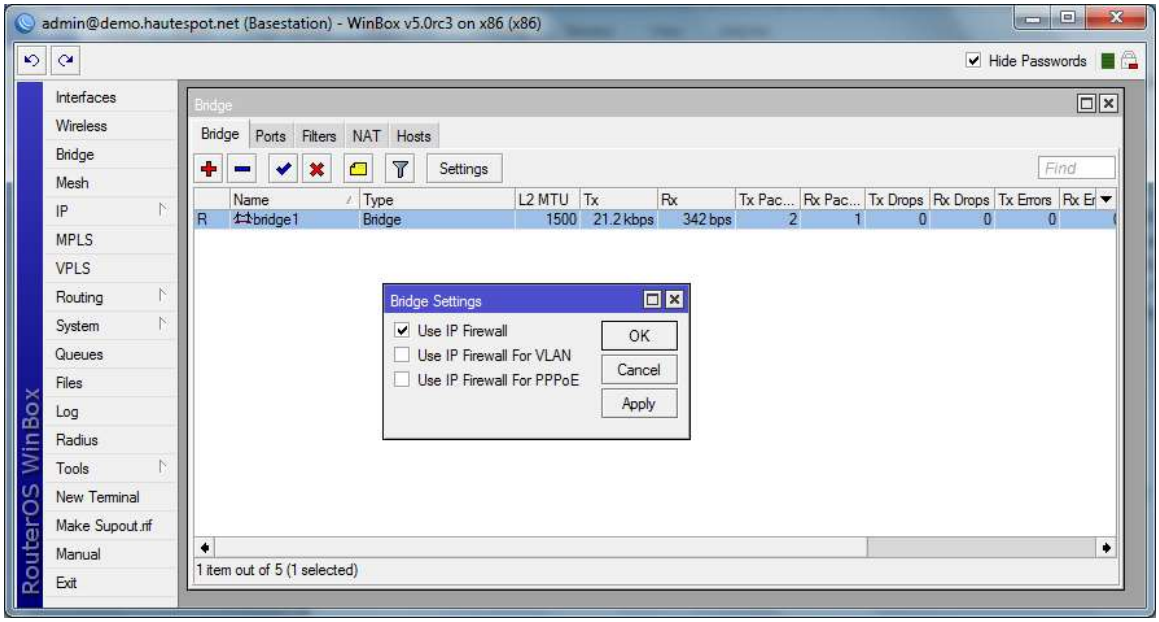
Properties

Property	Description
action (*accept* | *drop* | *jump* | *log* | *mark-packet* | *passthrough* | *return* | *set-priority*)	• **accept** - accept the packet. No action, i.e., the packet is passed through without undertaking any action, and no more rules are processed in the relevant list/chain • **drop** - silently drop the packet (without sending the ICMP reject message) • **jump** - jump to the chain specified by the value of the jump-target argument • **log** - log the packet • **mark** - mark the packet to use the mark later • **passthrough** - ignore this rule and go on to the next one. Acts the same way as a disabled rule, except for ability to count packets • **return** - return to the previous chain, from where the jump took place • **set-priority**

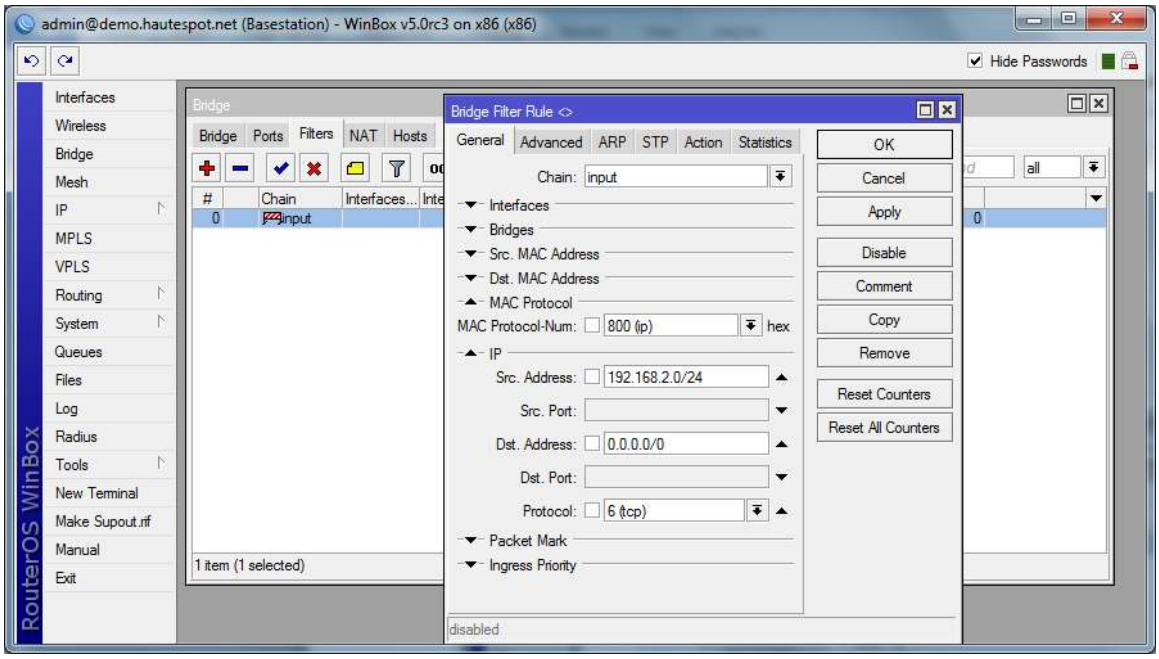
Example

A simple example to filter an IP source address on the bridge would be implemented as:

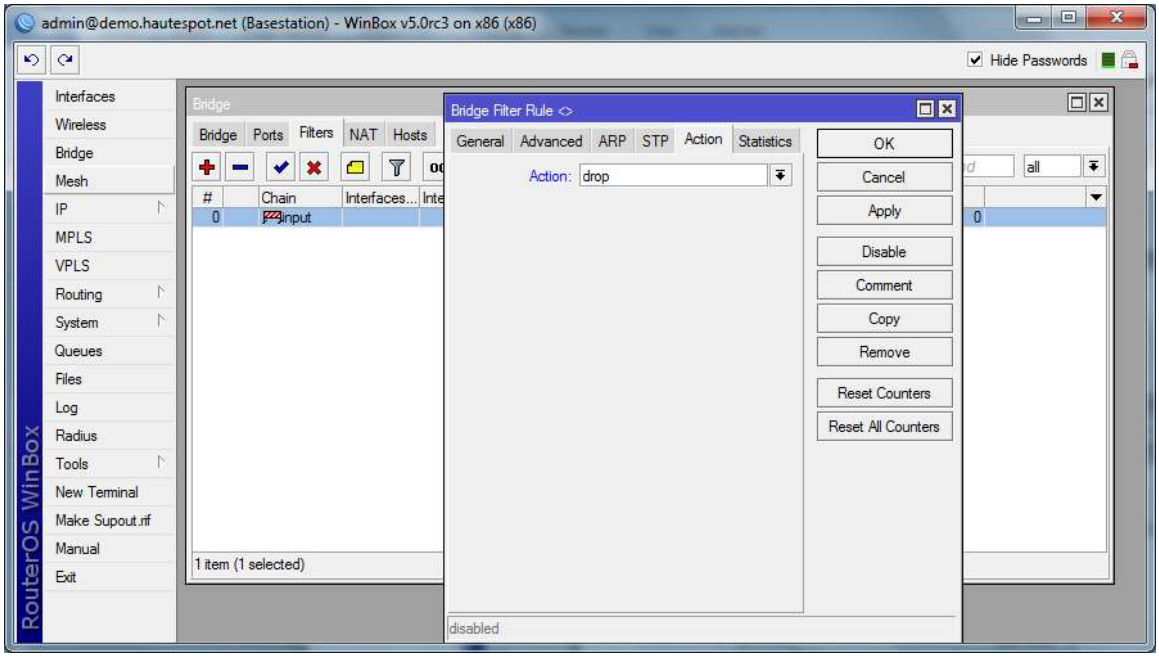
1. Enable IP Firewall on the bridge by going into bridge and click the settings button:



2. Then create a filter by selecting the filters tab and adding a new filter:



Select a chain of input, set the MAC Protocol to IP (800), and then specify the source network address with mask. Then switch to the Action tab and set the action to drop:



This would filter out and drop any traffic coming in from 192.168.2.0 network.

Filter rules are applied in order, top to bottom, so you can build permissive rules, denial rules etc.

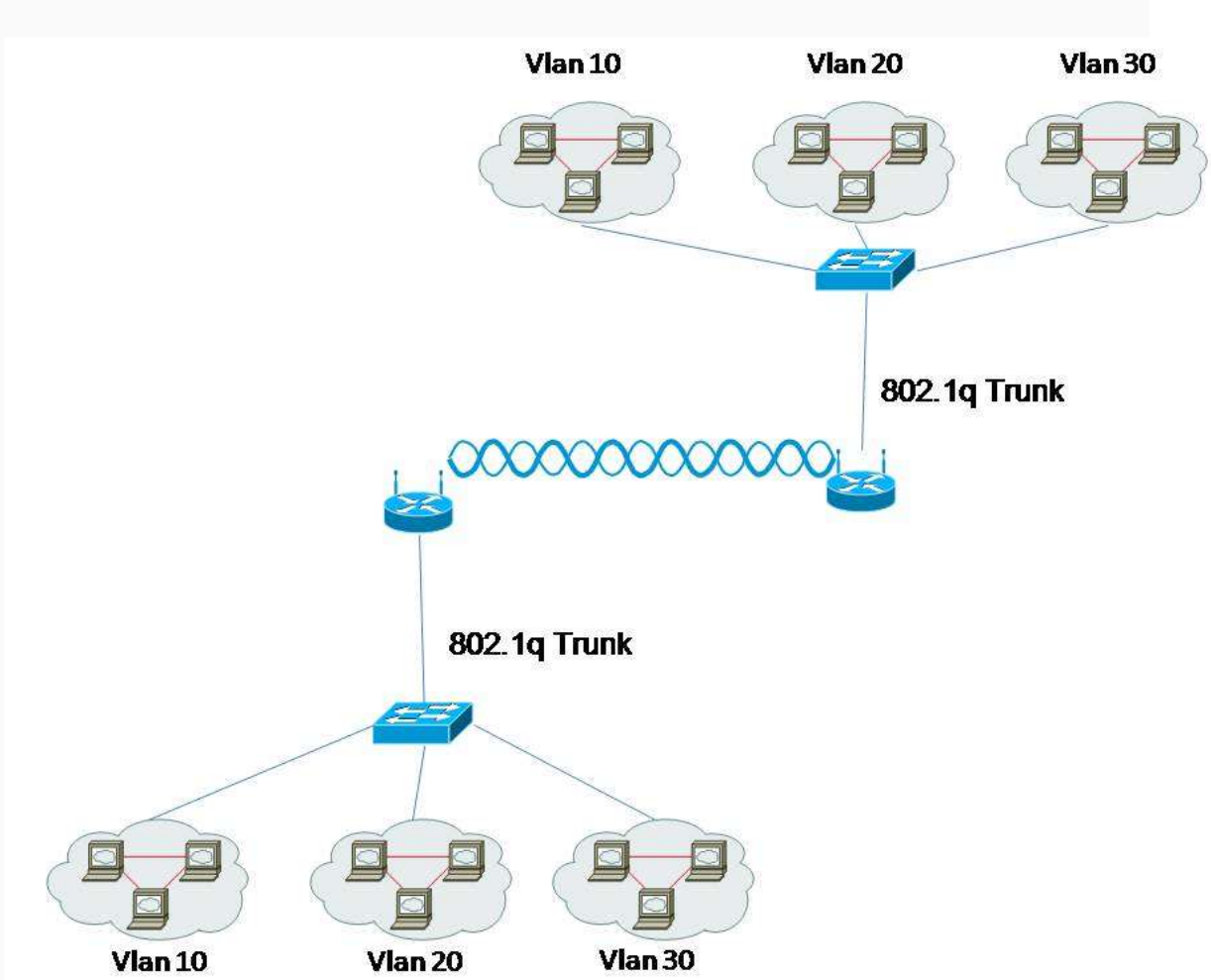
Creating Wireless VLANS

Introduction

It is very common practice to isolate group of users in a network using VLANs. In wired Networks we use lot of Trunk links to carry and extend VLANs from one switch to the other. Sometimes it becomes necessary to plan redundant parallel network links for these trunk links. Now building a trunk link using wi-fi network is little tricky.

Let us assume the Scenario as following.

- A network is having three VLANs with IDs -10,20,30
- A HauteSpot P2P link is being configured as trunk for carrying these three VLANs.
- Eth1 of the wireless routers are connected to the switches.

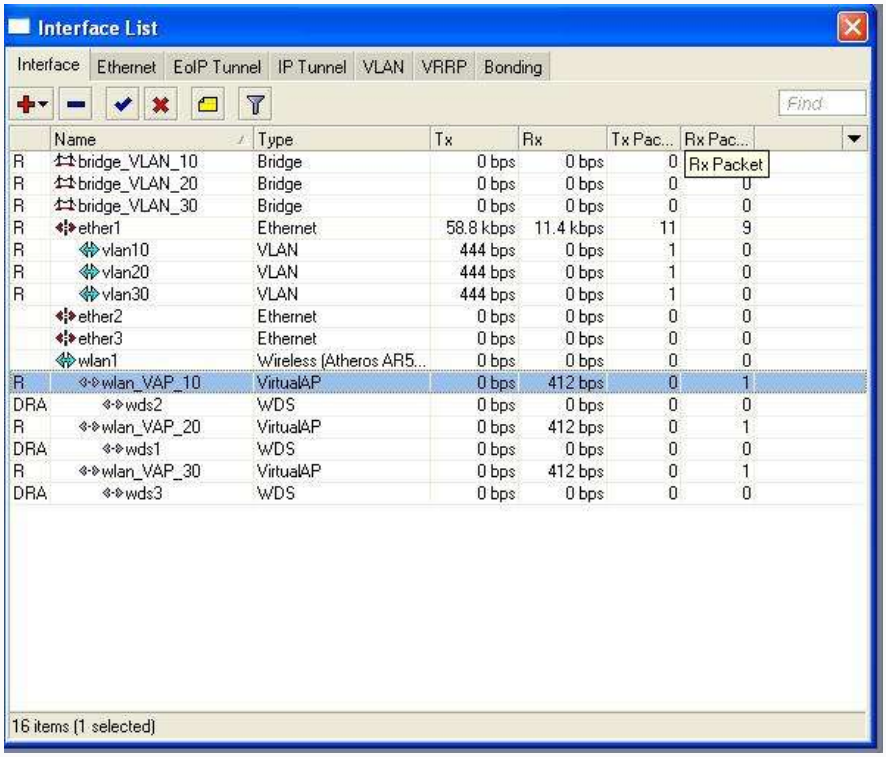


A. Create VLAN interfaces for your vlans

You need to create as many VLAN interfaces as the total no. of VLANs being forwarded through the trunk link. Here in this example I have maintained symmetry between VLAN ID and the name of the corresponding VLAN interface.

```
/interface vlan
add arp=enabled comment="" disabled=no interface=ether1 mtu=1500 name=vlan10 \
    vlan-id=10
add arp=enabled comment="" disabled=no interface=ether1 mtu=1500 name=vlan20 \
    vlan-id=20
add arp=enabled comment="" disabled=no interface=ether1 mtu=1500 name=vlan30 \
```

```
    vlan-id=30
```

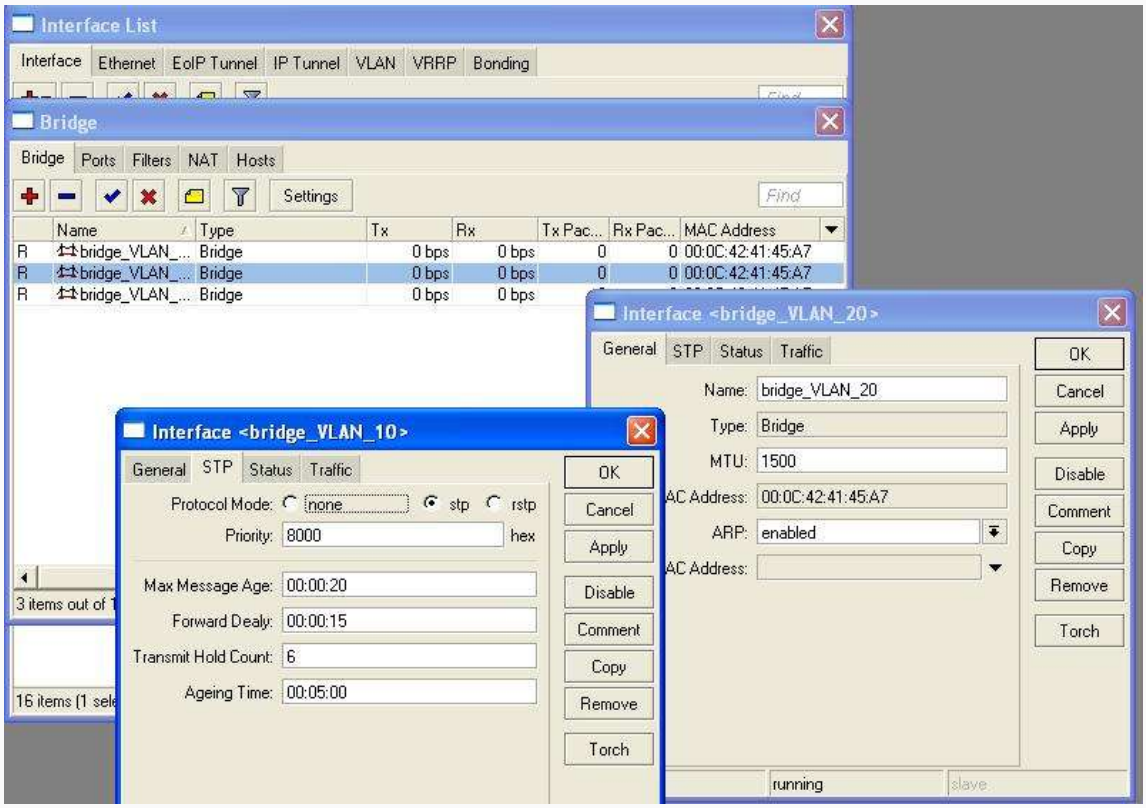

Interface List
Interface
Ethernet
EoIP Tunnel
IP Tunnel
VLAN
VRRP
Bonding
Find
Name
Type
Tx
Rx
Tx Pac...
Rx Pac...
Rx Packet
R bridge_VLAN_10 Bridge 0 bps 0 bps 0 0
R bridge_VLAN_20 Bridge 0 bps 0 bps 0 0
R bridge_VLAN_30 Bridge 0 bps 0 bps 0 0
R ether1 Ethernet 58.8 kbps 11.4 kbps 11 9
R vlan10 VLAN 444 bps 0 bps 1 0
R vlan20 VLAN 444 bps 0 bps 1 0
R vlan30 VLAN 444 bps 0 bps 1 0
ether2 Ethernet 0 bps 0 bps 0 0
ether3 Ethernet 0 bps 0 bps 0 0
wlan1 Wireless (Atheros AR5... 0 bps 0 bps 0 0
R wlan_VAP_10 VirtualAP 0 bps 412 bps 0 1
DRA wds2 WDS 0 bps 0 bps 0 0
R wlan_VAP_20 VirtualAP 0 bps 412 bps 0 1
DRA wds1 WDS 0 bps 0 bps 0 0
R wlan_VAP_30 VirtualAP 0 bps 412 bps 0 1
DRA wds3 WDS 0 bps 0 bps 0 0
16 items (1 selected)

B. Create bridges - one per vlan

One has to create as many bridges as the number of the vlan interfaces. This will run isolated Spanning Tree instances for the individual VLANs.

```
/interface bridge
add admin-mac=00:00:00:00:00:00 ageing-time=5m arp=enabled auto-mac=yes \
    comment="" disabled=no forward-delay=15s max-message-age=20s mtu=1500 \
    name=bridge_VLAN_10 priority=0x8000 protocol-mode=stp \
    transmit-hold-count=6
add admin-mac=00:00:00:00:00:00 ageing-time=5m arp=enabled auto-mac=yes \
    comment="" disabled=no forward-delay=15s max-message-age=20s mtu=1500 \
```

```
    name=bridge_VLAN_20 priority=0x8000 protocol-mode=stp \
    transmit-hold-count=6
add admin-mac=00:00:00:00:00:00 ageing-time=5m arp=enabled auto-mac=yes \
    comment="" disabled=no forward-delay=15s max-message-age=20s mtu=1500 \
    name=bridge_VLAN_30 priority=0x8000 protocol-mode=stp \
    transmit-hold-count=6
```

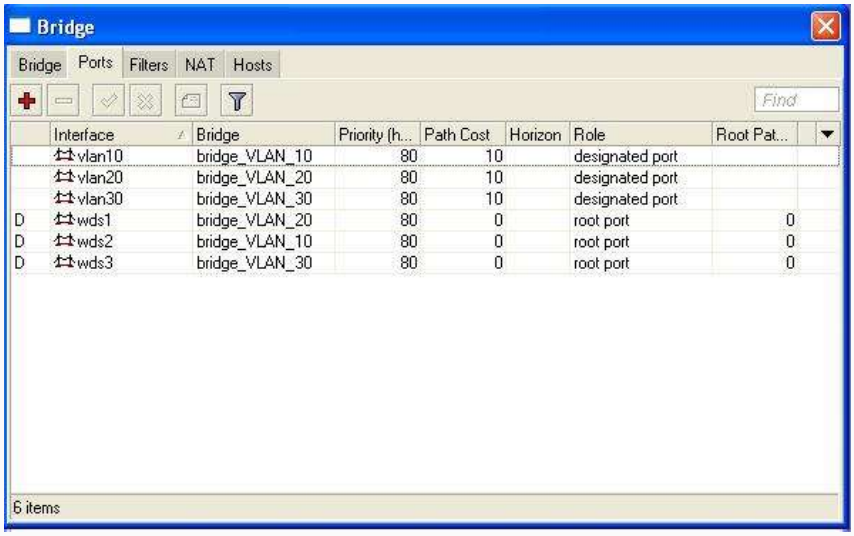


C. Add vlan interfaces to the corresponding bridge

After the creation of the bridges, vlan interfaces need to be assigned to the corresponding bridges. This is to note that no primary interface like ether1 or wlan1 is assigned to the bridges. If this is done by mistake, the whole system might stop forrwarding frames.

```
/interface bridge port
```

```
add bridge=bridge_VLAN_10 comment="" disabled=no edge=auto external-fdb=auto \
    horizon=none interface=vlan10 path-cost=10 point-to-point=auto priority=\
    0x80
add bridge=bridge_VLAN_20 comment="" disabled=no edge=auto external-fdb=auto \
    horizon=none interface=vlan20 path-cost=10 point-to-point=auto priority=\
    0x80
add bridge=bridge_VLAN_30 comment="" disabled=no edge=auto external-fdb=auto \
    horizon=none interface=vlan30 path-cost=10 point-to-point=auto priority=\
    0x80
```

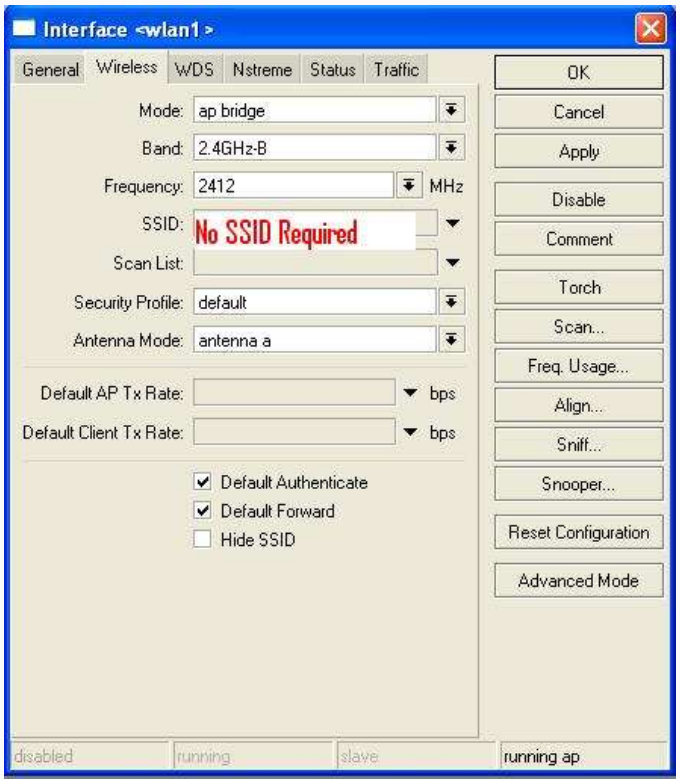


D. Create Virtual APs - one per vlan

- The Primary interface, wlan1 in this case will be in ap-bride mode and will not have any SSID associated with it.
- The virtual APs will have hidden SSIDs. While creating virtual APs put 00:00:00:00:00:00 as MAC address. RouterOS will dynamically assign a MAC address to these.
- In VAPs enable WDS in dynamic mode and select the corresponding VLAN bridge created earlier.

```
/interface wireless
```

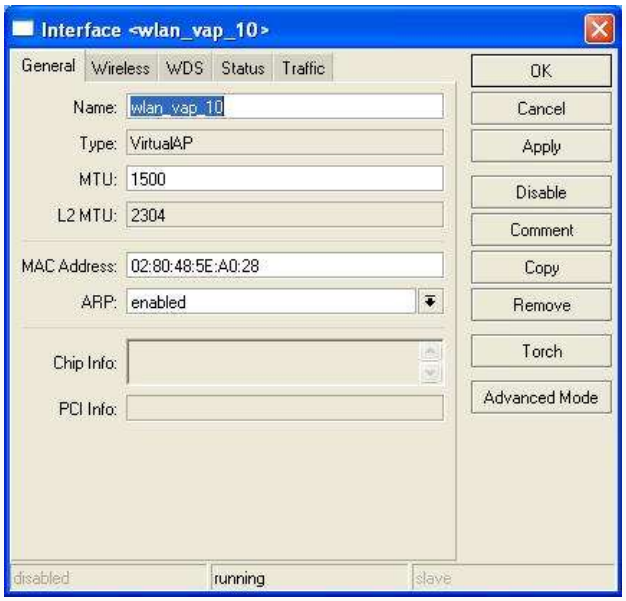
```
set 0 ack-timeout=dynamic adaptive-noise-immunity=none allow-sharedkey=no \
    antenna-gain=0 antenna-mode=ant-a area="" arp=enabled band=2.4ghz-b \
    basic-rates-a/g=6Mbps basic-rates-b=1Mbps burst-time=disabled comment="" \
    compression=no country=no_country_set default-ap-tx-limit=0 \
    default-authentication=yes default-client-tx-limit=0 default-forwarding=\
    yes dfs-mode=none disable-running-check=no disabled=no \
    disconnect-timeout=3s frame-lifetime=0 frequency=2412 frequency-mode=\
    manual-txpower hide-ssid=no hw-retries=4 mac-address=00:80:48:60:6B:91 \
    max-station-count=2007 mode=ap-bridge mtu=1500 name=wlan1 \
    noise-floor-threshold=default on-fail-retry-time=100ms \
    periodic-calibration=default periodic-calibration-interval=60 \
    preamble-mode=both proprietary-extensions=post-2.9.25 radio-name=\
    008048606B91 rate-set=default scan-list=default security-profile=default \
    ssid="" station-bridge-clone-mac=00:00:00:00:00:00 supported-rates-a/g=\
    6Mbps,9Mbps,12Mbps,18Mbps,24Mbps,36Mbps,48Mbps,54Mbps supported-rates-b=\
    1Mbps,2Mbps,5.5Mbps,11Mbps tx-power-mode=default update-stats-interval=\
    disabled wds-cost-range=50-150 wds-default-bridge=none wds-default-cost=\
    100 wds-ignore-ssid=no wds-mode=disabled wmm-support=disabled
```



Here in this example we have created three virtual APS associated with three different bridges through three separate WDS.

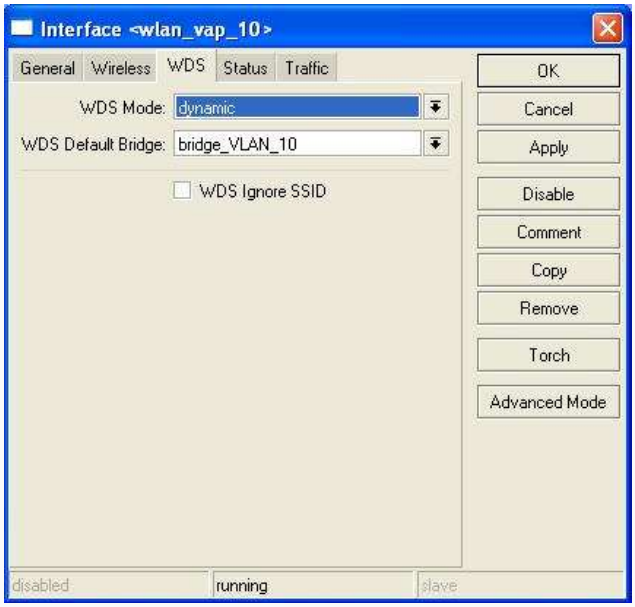
```
add area="" arp=enabled comment="" default-ap-tx-limit=0 \
    default-authentication=yes default-client-tx-limit=0 default-forwarding=\
    yes disable-running-check=no disabled=no hide-ssid=yes mac-address=\
    02:80:48:60:6B:91 master-interface=wlan1 max-station-count=2007 mtu=1500 \
    name=wlan_VAP_10 proprietary-extensions=post-2.9.25 security-profile=\
    default ssid=CISCO_V_10 update-stats-interval=disabled wds-cost-range=0 \
    wds-default-bridge=bridge_VLAN_10 wds-default-cost=0 wds-ignore-ssid=no \
    wds-mode=dynamic wmm-support=disabled
add area="" arp=enabled comment="" default-ap-tx-limit=0 \
    default-authentication=yes default-client-tx-limit=0 default-forwarding=\
     yes disable-running-check=no disabled=no hide-ssid=yes mac-address=\
     02:80:48:60:6B:92 master-interface=wlan1 max-station-count=2007 mtu=1500
\
     name=wlan_VAP_20 proprietary-extensions=post-2.9.25 security-profile=\
     default ssid=CISCO_V_20 update-stats-interval=disabled wds-cost-range=0 \
     wds-default-bridge=bridge_VLAN_20 wds-default-cost=0 wds-ignore-ssid=no \
     wds-mode=dynamic wmm-support=disabled
add area="" arp=enabled comment="" default-ap-tx-limit=0 \
     default-authentication=yes default-client-tx-limit=0 default-forwarding=\
     yes disable-running-check=no disabled=no hide-ssid=yes mac-address=\
     02:80:48:60:6B:93 master-interface=wlan1 max-station-count=2007 mtu=1500
\
     name=wlan_VAP_30 proprietary-extensions=post-2.9.25 security-profile=\
     default ssid=CISCO_V_30 update-stats-interval=disabled wds-cost-range=0 \
     wds-default-bridge=bridge_VLAN_30 wds-default-cost=0 wds-ignore-ssid=no \
     wds-mode=dynamic wmm-support=disabled
```

VAP General TAB: Assign a meaningful name to the Virtual AP.



VAP Wireless TAB: Create a meaningful SSID. It is advisable to hide SSID.

VAP WDS TAB: Turn on WDS in Dynamic mode and select the appropriate bridge.



Limitations

- The Aggregate Throughput of the trunk link will never be more that 22-23 Mbps using 802.11 or TLP in a 20MHz channel width without MIMO.
- Since one can create up to 128 Virtual APs, the total number of VLANs that this wireless link would be able to carry is 128 at maximum.
- This trunking will only work with 802.1q protocol. It is incompatible with ISL.

Questions and Answers

1. What about the AP configuration at the other end ? Is this article incomplete?

Since this is an end-to-end L-2 Set-up, both the APs would have identical configuration. Just get an export of the first AP and apply that on the second AP. Yes, it is that simple. The configuration is not incomplete. It does not have an IP address because this does not require one. Just following the steps described in this article, people were able to commission wireless trunks successfully.

2. Can this be used in point to multipoint?

Yes, if you put the remote ends into Station WDS mode. Doing this would allow each remote station to have it's own VLAN.
Alternatively you could use AP mode on each station, but the licenses of your stations need to support this. WRAPSTATION model equipment cannot be used in this way.

Option Globetrotter HSDPA USB Modem

Introduction

This example shows how to get the Option Globetrotter HSDPA USB Modem working with a generic carrier (UMTS/GPRS networks). Service in your country might have different requirements and strings, but this is just to give you an outline of what is required.

Hardware

USB Modem: Option N.V. Globetrotter HSDPA USB Modem H7.2

Modem manufacturer: Teltonika, model number U3G150

Router Hardware: HauteWRAP Router with USB port(s)

Router Software

HauteRouterOS V4.2 or later. Works in previous versions 2.9 and 3, but slightly different configuration.

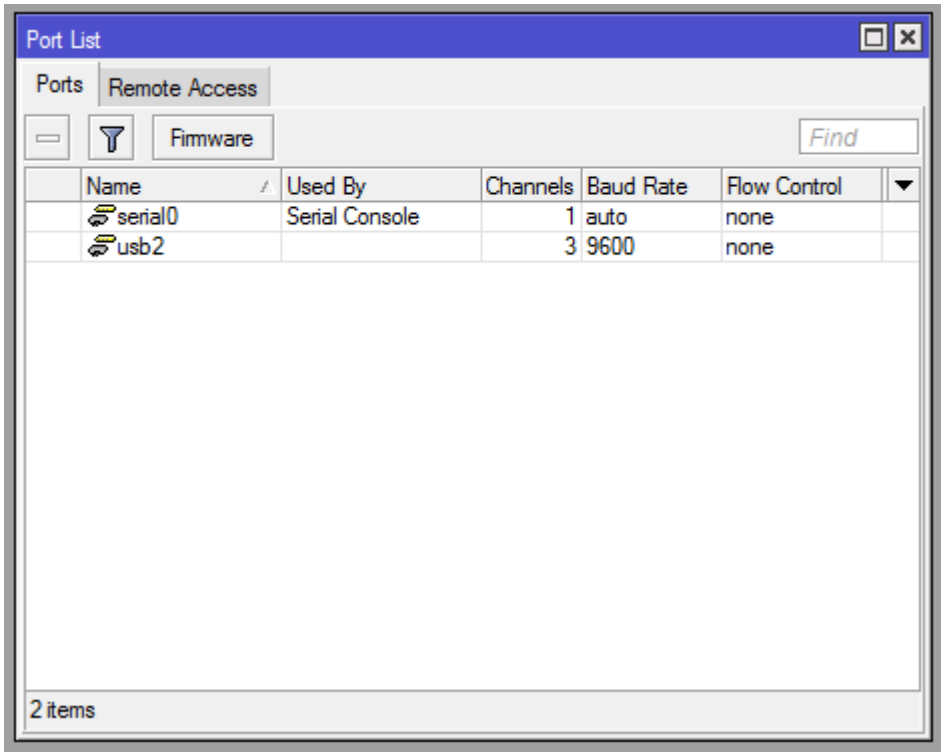
The USB Modem is recognized in HauteRouterOS as an USB device and listed under the USB resources:

```
[admin@hautespot] > /system resource usb print
 # DEVICE VENDOR                  NAME                   SPEED
2 1:3    Option N.V.              Globetrotter HSDPA... 12 Mbps
[admin@hautespot] >
```

Make sure the USB port is listed under the port menu:

```
[admin@hautespot] > /port print
Flags: I - inactive
 #   NAME                    CHANNELS   USED-BY                    BAUD-RATE
 0   serial0                 1          Serial Console             auto
 1   usb2                    3                                     9600
[admin@hautespot] >
```

In HS Configurator you can find this in the System Ports menu option:



Since HauteRouterOS V3.23 there is one port per modem, and modem has channels used for commands and data. Channels have numbers 0,1,2, etc. Some modems may have just two channels, some have more. Set the baud rate to 9600 for communicating with the modem, in case it's not done already:

```
[admin@hautespot] > /port set usb2 baud-rate=9600
```

Check the modem responses using serial-terminal, like this:

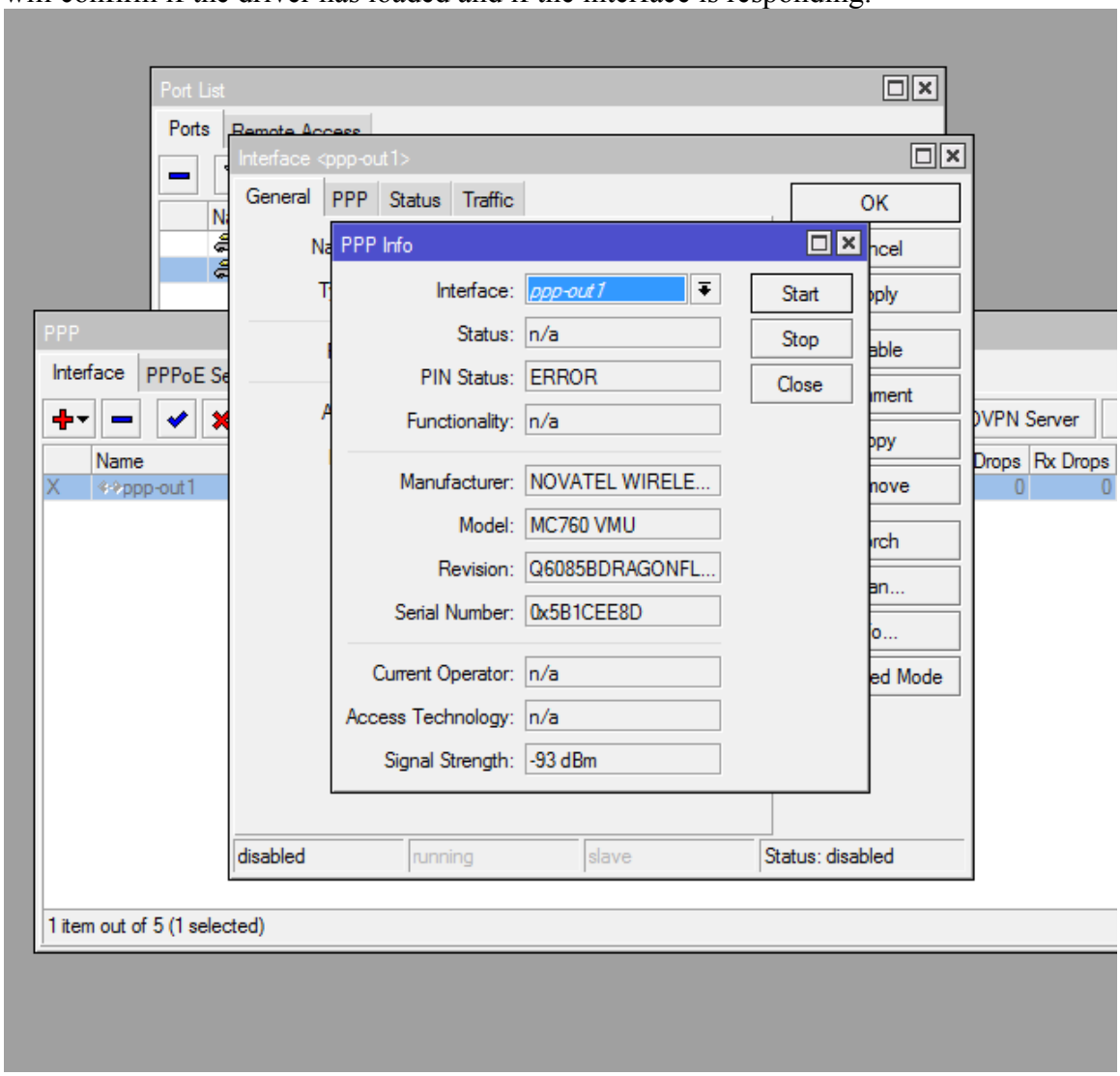
```
[admin@hautespot] > /system serial-terminal usb2 channel=2

[Ctrl-A is the prefix key]


ATI
Manufacturer: Option N.V.
Model: GTM378
Revision: 2.3.3Hd (Date: Jul 17 2007, Time: 15:49:23)

OK
```

Disconnect from the modem, by pressing "Ctrl-A", and then "Q":

```
[Q - quit connection]       [B - send break]
[A - send Ctrl-A prefix]    [R - autoconfigure rate]
```

```
Welcome back!
[admin@hautespot] >
```

In HS Configurator the easiest way to find the modem is to look in PPP interfaces where it should appear at ppp-out1. From this interface you can click on the info button and it will confirm if the driver has loaded and if the interface is responding:



Workaround for Globetrotter devices offering no modem interface

There is no guarantee that the commands above work on all the modems that need the HSO driver. The tested device was a T-Mobile branded Globetrotter iCON 225 "web'n'walk Stick". That device was able to connect without problems after the method described. Some Globetrotter HSDPA cards, for example the iCON 225 (branded as Orange, T-Mobile web'n'walk stick, etc. have the serial modem interface disabled by default, and only offer an NDIS network interface for the data connection. If a dial command is issued, but no CONNECT response happens from the modem, it might be helpful to check this. To check on the setting, issue:

```
[admin@hautespot] > /system serial-terminal port=usb2 channel=0

[Ctrl-A is the prefix key]


ATZ
OK
ATI0
Manufacturer: Option N.V.
Model: GlobeTrotter HSDPA Modem
Revision: 2.5.24Hd (Date: Apr 17 2009, Time: 08:59:36)

OK
AT_OIFC?
_OIFC: 2,1,1,0

OK
AT_OIFC=?

_OIFC: [2-3], [0,1], [1], [0-5]
         mdm,  diag,  app1,  pcsc/gps
MDM      (0: N/A)  (1: N/A) (2: ndis) (3: modem and ndis)
DIAG     (0: none) (1: enable)
APP1     (0: none) (1: enable)
PCSC/GPS (0: none) (1: pcsc enable) (2: GPS enable) (3: app2 enable) (4: GPS &
PCSC) (5: GPS & app2)
OK
```

In this example, the MDM (modem) interface is set to 2, so no serial modem interface is enabled in the device.

To change the setting, issue:

```
AT_OIFC=3,1,1,0
```

You should get: OK

To write the change permanently to the NVRAM of the device, issue:

```
AT&W
```

Unplug and re-plug the device, and issue:

```
[admin@hautespot] > /port print
Flags: I - inactive
 #   NAME                          CHANNELS USED-BY
BAUD-RATE
 0   usb1                                 4
9600
```

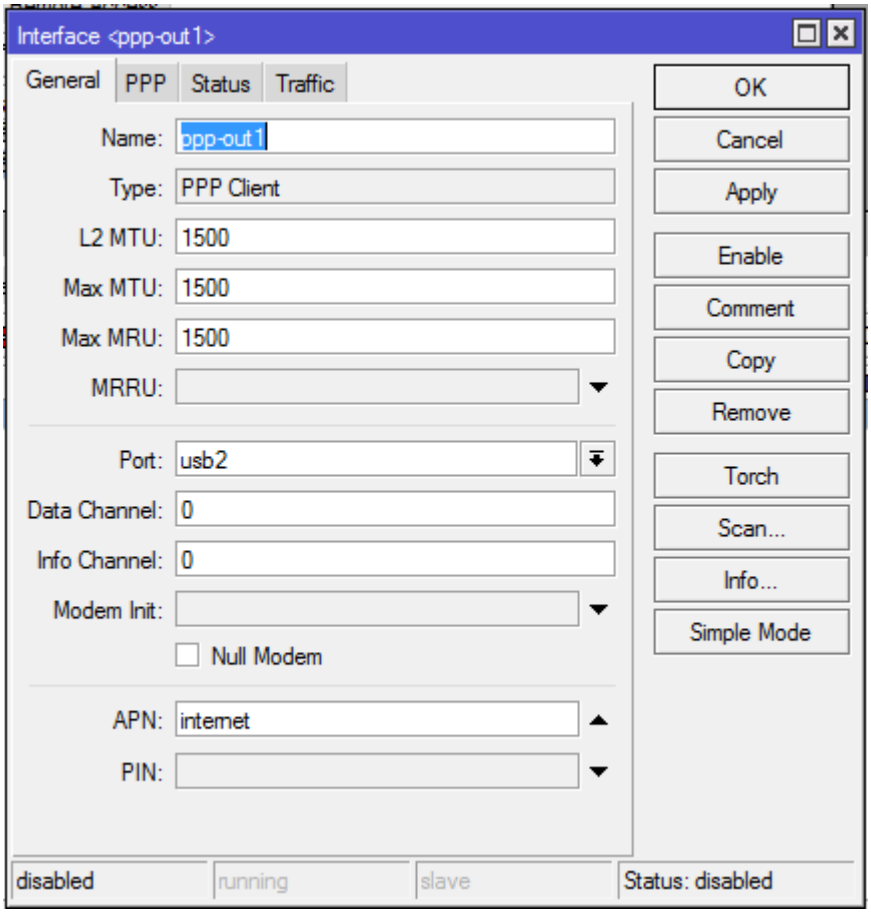
Number of channels should have increased. On the tested device, the last channel is the modem-interface (number 3 in HauteRouterOS).

According to the Linux HSO driver page at least five Option devices exist which do not have the modem interface enabled by default.

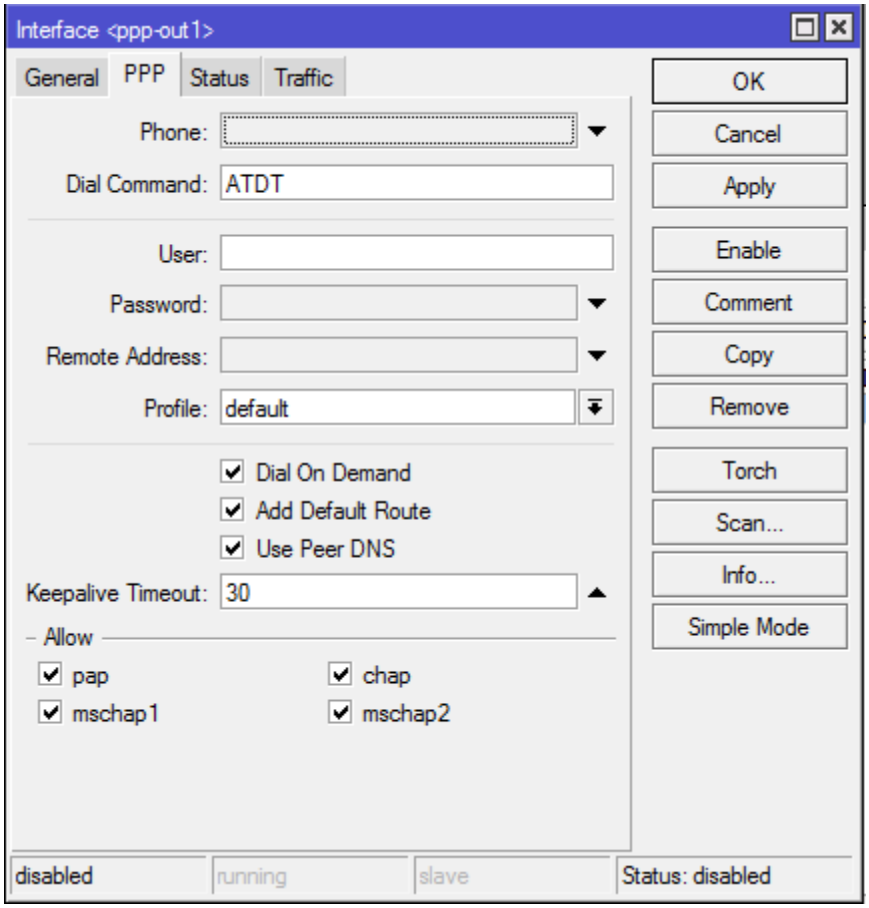
Router Configuration for PPP

Since RouterOS V4 (and latest versions 3) the PPP client configuration is made much easier compared to previous versions. There is no need to use ppp profiles, and ppp-client interface might have been added automatically to the /interface ppp-client list:

```
[admin@hautespot] > /interface ppp-client
[admin@hautespot] /interface ppp-client> print
Flags: X - disabled, R - running
 0 X  name="ppp-out1" max-mtu=1500 max-mru=1500 mrru=disabled port=usb2
      data-channel=2 info-channel=2 apn="internet" pin="" user=""
password=""
      profile=default phone="" dial-command="ATDT" modem-init=""
      null-modem=no dial-on-demand=yes add-default-route=yes use-peer-
dns=yes
      allow=pap,chap,mschap1,mschap2
[admin@hautespot] /interface ppp-client> info 0
```



The interface has 3G/GPRS modem specific arguments, like apn, pin. Consult your network provider what to use for APN, user, and password. Adjust your peer-dns and default-route settings according to what you want to do over that interface! In our case, we want permanent connection and won't use dial on demand:




```
[admin@hautespot] /interface ppp-client> set 0 dial-on-demand=no
[admin@hautespot] /interface ppp-client> enable 0
[admin@hautespot] /interface ppp-client> print
Flags: X - disabled, R - running
 0  R name="ppp-out1" max-mtu=1500 max-mru=1500 mrru=disabled port=usb2
      data-channel=2 info-channel=2 apn="internet" pin="" user="" password=""
      profile=default phone="" dial-command="ATDT" modem-init=""
      null-modem=no dial-on-demand=no add-default-route=yes use-peer-dns=yes
      allow=pap,chap,mschap1,mschap2
[admin@hautespot] /interface ppp-client>
```

After enabling the interface watch the logs to see how the connection is being established over the USB modem:

```
[admin@hautespot] /interface ppp-client> /log print
12:29:16 async,ppp,info ppp-out1: initializing...
12:29:16 async,ppp,info ppp-out1: reseting link...
12:29:16 system,info device changed by admin
12:29:16 system,info dns changed
12:29:17 async,ppp,info ppp-out1: initializing modem...
12:29:17 async,ppp,info ppp-out1: dialing out...
12:29:17 async,ppp,info ppp-out1: authenticated
12:29:20 async,ppp,info ppp-out1: could not determine remote address, using
10.112.112.119
12:29:20 async,ppp,info ppp-out1: connected
```

```
12:29:20 system,info dns changed

[admin@hautespot] /interface ppp-client>
```

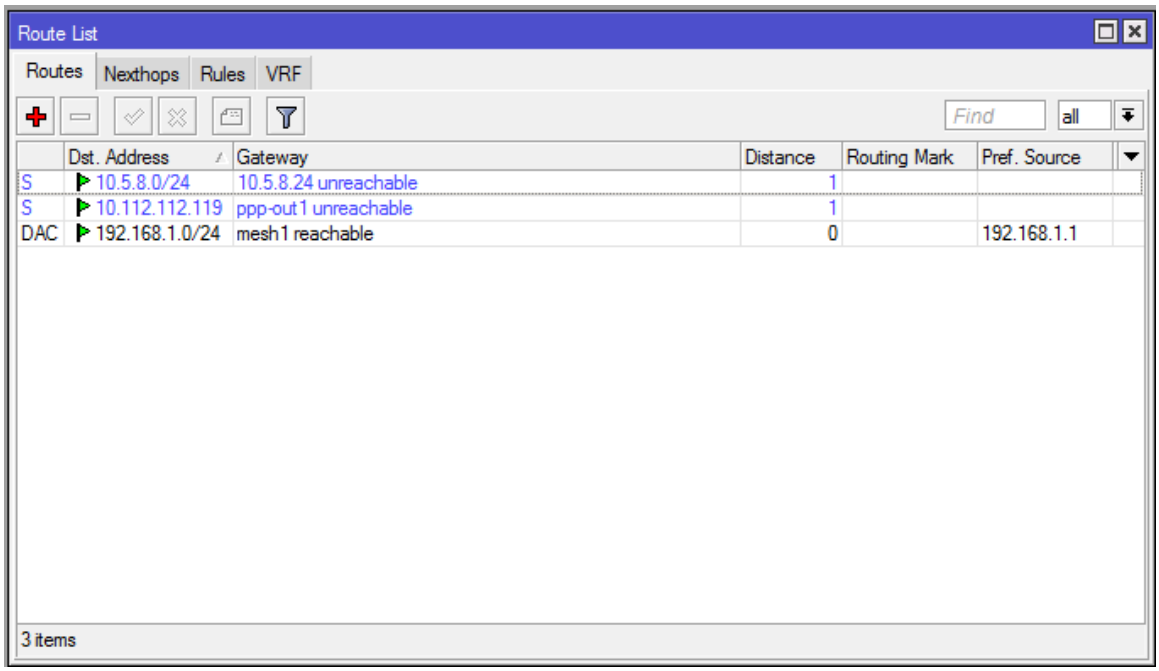
The serial port is used by ppp client:

```
[admin@hautespot] > /port print
Flags: I - inactive
 #    NAME                      CHANNELS   USED-BY                      BAUD-RATE
 0    serial0                   1          Serial Console               auto
 1    usb2                      3          PPP <ppp-out1>              9600
[admin@hautespot] >
```

Check the addresses, in our case we have:

```
[admin@hautespot] > /ip address print
Flags: X - disabled, I - invalid, D - dynamic
 #    ADDRESS            NETWORK         BROADCAST       INTERFACE
 0 D 10.5.8.64/24       10.5.8.0        10.5.8.255      ether1
 1 D 10.40.192.214/32   10.112.112.119  0.0.0.0         ppp-out1
[admin@hautespot] >
```

and routes:




```
[admin@hautespot] > /ip route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        GATEWAY             DISTANCE
 0 ADS  0.0.0.0/0                          10.112.112.119      1
 1 ADC  10.5.8.0/24        10.5.8.64       ether1              0
```

```
 2 ADC  10.112.112.119/32  10.40.192.214   ppp-out1           0
[admin@hautespot] >
```

You can get more status information about the modem and network by running the "info" command. In our case the modem uses the same channel for data and info, thus, we disable the ppp-client and run the "info" command then:

```
[admin@hautespot] /interface ppp-client> info 0
failure: can't reuse channel while ppp-client running!
[admin@hautespot] /interface ppp-client> disable 0
[admin@hautespot] /interface ppp-client> info 0
             status: "ready"
         pin-status: "no password required"
      functionality: "minimum"
         gprs-class: "A - GPRS & GSM simultaneous"
       manufacturer: "Option N.V."
              model: "GTM378"
           revision: "2.3.3Hd (Date: Jul 17 2007, Time: 15:49:23)"
      serial-number: "356237010662581,S23977B51N"
   current-operator: "LV LMT GSM"
  access-technology: "3G"
     signal-strengh: -89

[admin@hautespot] /interface ppp-client>
```

You may need to use masquerade to hide your private network when going out through the ppp!

Modem Preferred Operation Settings

OPTION modems can be set for preferred operation in UMTS(3G) or GPRS networks. To change the operation mode, use AT_OPSYS command via terminal, or, include it into the modem init string:

```
AT_OPSYS=0,2 Only GPRS
AT_OPSYS=1,2 Only UMTS(3g)
AT_OPSYS=2,2 Prefer GPRS over UMTS(3G)
AT_OPSYS=3,2 Prefer UMTS(3G) over GPRS
```

Query the current _OPSYS setting using the AT_OPSYS? command:

```
[admin@hautespot] > sys serial-terminal usb2 channel=0

[Ctrl-A is the prefix key]


AT_OPSYS?
_OPSYS: 3,2

OK
```

The _OPSYS operation mode setting stays intact after router reboot or power off/on. If you lock the modem operation setting to 3G only, the scan command won't show GPRS networks, and vice versa.

Troubleshooting

- If you do not get "authenticated" and "connected", then there is something wrong with your settings. Most likely, the ppp-client interface settings should be changed to match your provider's requirements.

- If the data-channel setting is wrong, there would be no connection. The number of available channels is shown under the /port list, port numbers start with 0. Disable the interface and try another port number!

- Check if you can communicate with the modem by using the serial-terminal! Make sure the ppp-client is disabled!

- Turn off PIN request for your SIM card, it makes the life much easier. Do it in a phone if you do not know the correct AT command.

- When speed of the link is not what expected, it's possible that the modem has connected to GPRS network, not 3G. Use the "scan" command for list of available networks:

```
[admin@hautespot] /interface ppp-client> scan 0
Flags: C - current, A - available, F - forbidden
  OPERATOR                    CODE       ACCESS-TECHNOLOGY
SIGNAL-STRENGH
C LV LMT GSM                  24701      GSM compact
-65
A LV LMT GSM                  24701      3G
F BITE LV                     24705      3G
F LV TELE2                    24702      GSM compact
F LV TELE2                    24702      3G
F BITE LV                     24705      GSM compact
-- [Q quit|D dump|C-z pause]
```

- If you need to add multiple inits (multiline inits) then it is possible by dividing them using ; character. Example:

```
modem-
init="AT+CGDCONT=5,\"IP\",\"internet.lmt.l\";AT+CGDCONT=6,\"IP\",\"inte
rnet.lmt.lv\";AT"
```

Typical things to check

- sms sending not working

1. check port
2. check PIN
3. enable debug logging GSM, DEBUG
4. check log
5. create supout

- sms receiving not working

1. check port
2. check PIN
3. got delivery accounting sms ?
4. allowed-number is correct with country code prefix +XXX ?
5. secret matches ?
6. check sms store directly /sys serial usbX by issuing command at+cmgl=1 (lists all messages in message store)
7. enable debug logging GSM, DEBUG
8. check log
9. create supout

- ppp client not working

1. check port
2. check PIN
3. dial-on-demand is set to no?
4. apn correct?
5. needs special modem-init (to handle PIN use at+cpin=XXXX) ?
6. enable debug logging with topics ASYNC, DEBUG
7. check log
8. create supout

- ppp info not working

1. check port
2. uses same channel info-channel == data-channel (can't work while ppp-client is enabled) ?
3. enable debug logging with topics ASYNC, DEBUG
4. check log
5. create supout

- ppp info command reports functionality=minimum

1. use /system serial-terminal usbX channel=X and execute the at+cfun=1

2. ppp info command should report functionality=full
3. If after each reboot the functionality is in minimum state add the at+cfun=1 into the modem-init field

- after executing some AT commands the modem doesn't respond correctly or reports error values

1. resetting the modem functionality might solve the problem
2. use /system serial-terminal usbX channel=X
3. execute the at+cfun=0 command and wait few seconds and execute at+cfun=1

- port invalid or not working

1. unplug the modem
2. disable the ppp-client interface
3. reboot the router
4. plug in the modem and after 5s proceed to nex step
5. create supout

- modem stops responding

1. upgrade the modem firmware and check if it solves the problem
2. create supout

To check port

- port busy ? /port print (check used-by)
- channel busy ? (two application can't use same channel)
- is chosen channel working? check if you get any output /sys ser usbX channel=X with command ATI
- port shown ? /port print
- device shown ? /sys reso usb pr
- create supout

To check PIN

- /int ppp-client info CONNAME or run /sys serial usbX and issue at+cpin?
- enter pin at+cpin=XXXX

To create supout

- run /sys sup-output
- contact support and provide supout file

3G modem cards supported

Model	Tested Haute RouterOS version	Comments	Format
AirPrime/Sierra PC 5220!	v3		PCMCIA
Alcatel One Touch X020X USB (aka Longcheer WM66; Nuton NT36HD; MWalker mbd 100hu; Novacom GNS-3.5G White, SU-8200U; MTE MW610?)	v5.10	Config like Option_Globetrotter_HSDPA_USB_ Modem Connected to Internet, Did not test Speed + Reliability (Alcatel OT X020X on x86) (data 0, info channel: 2)	USB
AnyData ADU E100A (aka "USB Wireless HSDPA/UMTS 2.1GHz GSM/GPRS/EGPRS 900/17000MHz/CDMA 1x EVDO Rev.A")	v3.14		USB
AnyData ADU 500A USB (aka "USB Wireless HSDPA/UMTS 2.1GHz GSM/GPRS/EGPRS 900/1800MHz/CDMA 1x EVDO Rev.A")	v3.14		USB
Audiovox PC5220 CDMA Dual Band 1XEV-DO PC Card	v3		PCMCIA
C-motech CNU-680 CDMA 1x EV-DO 450Mhz USB Modem (used by Triatel) [1]	v3.22		USB
Dell 5520	v3		
Ericsson_F3507g_Mobile_Broadband_Module [2]	V3.28	Set init string AT+CFUN=1, data channel and info channel to 3.	MiniPCI-e
Huawei E226 USB modem,	3.25		USB
Huawei E220 USB modem,	3.0rc8		USB

Model	Tested Haute RouterOS version	Comments	Format
E200BIS [3]			
Huawei E169 USB modem (used by Tele2) [4]	3.22		USB
Huawei E180 USB modem	3.20		USB
Huawei E1550 [5]	v3.26		USB
Huawei E1553 (USB)	v4.6		USB
Siemens M20	v3.28		
Huawei Mobile Connect Model E620 (3G);	v3		
Kyocera KPC650	v3		
Nokia CS-17 (USB)	v5.0	data channel=2, info channel=4	USB
Nokia CS-18 (USB) Rogers Canada	5.12	Data channel=1, Info channel =1, APN= internet.com, Phone = *99# , Dial = ATDT , pap , Tested on rb-751 and rb-493	USB
Novatel EU740	v3		MiniPCI-e
Novatel EU870 [6]	v3		
Novatel MIFI 2372 Bell Canada	5.12	Data Channel=0 , Info Channel= 0, APN = pda2.bell.ca , Phone = *99# , Dial = ATDT , pap, Tested on Rb-750UP and RB-493	USB
Novatel EV620 CDMA/EV-DO	v3		MiniPCI-e
Novatel Merlin ES620 / Merlin ES720 / Ovation U720 [7]	v3		USB
Novatel Merlin ES620 SM Bus	v3		
Novatel Merlin S720 (HSDPA) [8]	v3		PCMCIA
Novatel Merlin XU870 HSDPA/3G [9]	v3		ExpressCard
Novatel U720 Wireless CDMA Modem	v4.5		USB

Model	Tested Haute RouterOS version	Comments	Format
Novatel U730 (Wireless HSDPA Modem) [10]	v3		PCMCIA
Novatel Wireless CDMA card	v3		
Option Fusion UMTS Quad-GPRS (Vodafone Globetrotter);	v3.4		
Option Globetrotter HSDPA USB aka Teltonika ModemUSB/H7.2 (U3G150) [11]	v3		USB
Probably most of the cards needing the HSO driver on Linux, tested: Option Globetrotter HSDPA USB (Globetrotter iCon 225 [12]	v5.12	Option_Globetrotter_HSDPA_USB_Modem see Workaround for Globetrotter devices offering no modem interface	USB
Option Qualcomm 3G WCDMA Model M00201-10886 (GTM378) [13]	v3.27		miniPCI-e
Option Qualcomm 3G CDMA Model M00301 (GTM380) [14]	v3.28	Set data channel and info channel to 3.	miniPCI-e
Option Qualcomm 3G CDMA Model M00401 (GTM382) [15]	v3.28 & v4.6	Set data channel and info channel to 3.	miniPCI-e
Ericsson 3G F3607gw miniPCI-e	v3.28	Set data channel and info channel to 2. Set init string AT+CFUN=1	miniPCI-e
Sierra Aircard 595 [17]	v3		PCMCIA
Sierra Aircard 595U USB Sprint Card [18]	v3		USB
Sierra Wireless USB 306	v5.9	Data & Info Channel 2. For Telecom NZ use APN internet.telecom.co.nz and Phone number *99#	
Sierra Wireless USB 308 or AT&T Shockwave [19]	v5.0rc11	AT Commands are sent through Data Channel 2 or 3. Change this under Interface -> PPP, General Tab	USB

Model	Tested Haute RouterOS version	Comments	Format
		using Advanced Mode, or command line /interface ppp-client set 0 data-channel=2	
Sierra Wireless AirCard 312U [20]	v5.2	Use Info Channel 2 and Data Channel 3. Modem also known in Australia as the Telstra BigPond Ultimate. The MicroSD slot on the modem can be used for additional storage (tested on 5.11) such as a web proxy.	USB
Sierra Wireless AirCard 580 [21]	v3		PCMCIA
Sierra Wireless AirCard 595 [22]	v3		PCMCIA
Sierra Wireless AirCard 597E [23]	v3		ExpressCard
Sierra Wireless AirCard 875 [24]	v3		PCMCIA
Sierra Wireless AirCard 880 [25]	v3.8		PCMCIA
Sierra Wireless AirCard 880 E [26]	v3.8		ExpressCard
Sierra Wireless AirCard 881 [27]	v3.8		PCMCIA
Sierra Wireless AirCard 881 E [28]	v3.8		ExpressCard
Sierra Wireless EM5625 [29]	v3		MiniPCI-e
Sierra Wireless MC5720 [30]	v3		MiniPCI-e
Sierra Wireless MC5725 [31]	v3		MiniPCI-e
Sierra Wireless MC8705 [32]	v5.1	Data and Info channel are 3. Suitable for 850,900 and 2100Mhz 3G networks. GPS not yet tested. Sim card slot required. Capable of HSPA+ speeds of 21Mbps down and 5.76Mbps up.	MiniPCI-e

Model	Tested Haute RouterOS version	Comments	Format
Sierra Wireless MC8755 [33]	v3	Users have reported that the device doesn't show up after shutdown, reboot is required. Init-string: ATE0V1&D2&C1S0=0+IFC=2,2;AT+CGDCONT=1,"IP","apn-string"; - where the apn-string is your correct APN. A reboot can be required after settings are changed, to make it working. Data and Info channel are 0.	MiniPCI-e
Sierra Wireless MC8755 for Europe [34]	v3		MiniPCI-e
Sierra Wireless MC8765 [35]	v3		MiniPCI-e
Sierra Wireless MC8775 [36]	v3		MiniPCI-e
Sierra Wireless MC8780 [37]	v3.8		MiniPCI-e
Sierra Wireless MC5725 [38]	v3.18		MiniPCI-e
Sierra Wireless MC5727 [39]	v3.18		MiniPCI-e
Sierra Wireless MC8785	v3.18		MiniPCI-e
Sierra Wireless MC8790 [40]	v3.18	Few models do not send echo for input commands, modem does not work properly.	MiniPCI-e
Sierra Wireless MC8792 [41]	v5.2	Info works only in channel 3. Channels 4 and 5 has limited AT set. datachannel=4, infochannel=3	MiniPCI-e
Sierra Wireless MC8781 [42]	v3.8		MiniPCI-e
Sierra Wireless Sierra 598 (Sprint) USB [43]	v3.x		USB
Sierra Wireless MP3G - EVDO	v3.8		
Sierra Wireless MP3G - UMTS/HSPA	v3.8		
Sierra Wireless Compass 885 (USB) [44]	v3.28	AKA Telecom New Zealand's new XT network. Works a treat.	USB
Silicon Labs MobiData GPRS USB Modem	v3.22		USB

Model	Tested Haute RouterOS version	Comments	Format
Sprint U301/301U 4G wireless card [45]	v4.6	C-MOTECH Co, FW301DOWMX, QUALCOMM Patch 3350. Data CH=1 Info CH=3 Phone #777 for Sprint in US	USB
Sprint U300/300U 4G wireless card [46]	v4.6	C-MOTECH Co, FW301DOWMX, QUALCOMM Patch 33504	USB
Franklin M600 3G/4G wireless card [47]	v5.x	only 3G mode works	MiniPCI-e
Verizon Express Network PC5220 (AirPrime 5220)	v3		
ZTE AC8700	v3.6		
ZTE MF620 / MF622 [48]	v3		USB
ZTE MF620 / MF622 (3G) [49]	v3.6		USB
ZTE MF680 [50]	v5.4	Used by 3 in Sweden. Set data chanel to 1	USB
ZTE MF668 [51]	v4.5	for Rogers Wireless (Canada) Set APN: isp.apn and Info & Data Channel to 1	USB
T-Mobile (Germany) Web´n´Walk Box Micro (Huawei E220) [52]	v3.x		USB
Vodafone (Germany) Easybox 2 (Huawei E220) [53]	v3.x		USB
O2 (Germany) Surfbox Mini (Huawei E220) [54]	v3.x		USB
E-Plus & Base (Germany) USB Minimodem (Huawei E220) [55]	v3.x		USB
Huawei E600	v3.15		
Novatel Merlin V640/XV620 [56]	v3.15		ExpressCard
Novatel Merlin V620/S620 [57]	v3.15		PCMCIA

Model	Tested Haute RouterOS version	Comments	Format
Novatel Merlin EX720/V740/X720 [58]	v3.15		ExpressCard
Novatel Merlin V720/S720/PC720 [59]	v3.15		PCMCIA
Novatel Merlin XU870 HSDPA/3G [60]	v3.15		ExpressCard
Novatel X950D [61]	v3.15		ExpressCard
Novatel ES620/ES720/U720/USB720	v3.15		USB
Novatel E725/E726 [62]	v3.15		MiniPCI-e
Vodafone EU740/Novatel non-Vodafone EU740	v3.15		MiniPCI-e
Vodafone K3565/Huawei E160 [63]	v3.23		USB
Novatel EU850D/EU860D/EU870D [64]	v3.15		MiniPCI-e
Novatel MC930D/MC950D [65]	v3.15		USB
Novatel MC727/U727 [66]	v3.15		USB
Novatel Expedite EV620 CDMA/EV-DO	v3.15		MiniPCI-e
Novatel Expedite EU740 HSDPA/3G, Dell Wireless 5500 Mobile/Dell Wireless 5505 Mobile	v3.15		MiniPCI-e
Novatel Expedite E720 CDMA/EV-DO	v3.15		MiniPCI-e
Novatel Expedite ET620 CDMA/EV-DO	v3.15		
Onda H600/ZTE MF330	v3.15		
BP3-USB & BP3-EXT HSDPA	v3.15		USB
ZTE MY 39 (MSM 6500	v3.20		PCMCIA

Model	Tested Haute RouterOS version	Comments	Format
based) [67]			
Cricket A600	v3.26		
Globetrotter HSDPA Modem Option N.V.	v3.26		
Sony Ericsson MD300	v3.26		
ZTE MF 626 [68]	v3.26		USB
ZTE MF 627 [69]	v3.26		USB
Pantech / UTStarcom UM175	v3.27	On Alltel's CDMA EVDO RevA network average 700-800kbps up and 2.4-2.8Mbps down	
Novatel U760 [70]	v3.30		USB
ZTE K3565-Z [71]	v4.4	Revision: BD_P673A2V1.0.0B09	USB
Novatel Expedite EV620	v4.5		MiniPCI-e
Novatel MC760 VMU [72]	v4.5		USB
Franklin Wireless FW300DOWMX	v4.5		
Huawei EC1260	v4.5		
Vodafone K3520-Z [73]	v4.6		USB
Vodafone K3765 [74]	v4.6		USB
Telstra 3G Elite	v5.x		
Vodafone Huawei K4505	v5.x	Data-channel=0 Info-channel=3	USB
Vertex VW 110	v5.x		
ZTE MF112 [75]	v5.x	Power issues on mipsbe boards	USB
Huawei ET127	v5.x		
Huawei EC1261	v5.x		
Huawei E173 [76]	v5.x		USB
ZTE MF190 [77]	v5.x	Data channel=3 and info Channel=1	USB
ZTE MF102 [78]	v5.x	Possible that need to change data channel=2 and info channel=2	USB
China TeleCom or ZTE AC682	v5.x		
Option Globetrotter GT380	v5.x		

Model	Tested Haute RouterOS version	Comments	Format
Simcom 5220	v5.x		
Huawei K3770	v5.x		
Novatel USB551L (Verizon) [79]	v5.9	Only 3G support (No LTE support)	UB
Novatel Wireless MIFI4510		Not supported	
ZTE MC2718 [80]	v5.8	Possible data-channel=0 info-channel=1 GPS(NMEA)=4	MiniPCI-e
LG-VL600 (Verizon)		Not supported	
Huawei EC156 [81]	v5.8		USB
K3806 [82]	v5.8		USB
ZTE MF-210V [83]	v5.9		MiniPCI-e
Huawei E398 [84]	v5.9	2G, 3G, 4G supported	USB
Huawei E367	v5.11	Works OK	USB
Huawei EM770 [85]	v5.11	interface listing at the startup might be slowed down for a small period of time	MiniPCI-e
Nokia E52 (Series 60)	v5.12	Set Usb mode to "PC Suite" in phone menu Baud rate - 115200 Ports - 0/0 APN - internet No modem init string, Dial number - *99# Phone is charged up through the USB port	USB
Nokia 6700 Classic (Series 40)	v5.12	Set Usb mode to "PC Suite" in phone menu Baud rate - 115200 Ports - 0/0 APN - internet Modem init string - ATZ, Dial number - *99# Phone is charged up through the USB port	USB
Alcatel X220S [86]	v5.13		USB

Model	Tested Haute RouterOS version	Comments	Format
MO835UP	v5.14		USB
Nokia Datacard CS-11 & CS-15	v5.14		USB
Pantech UML290	V5.12	3G EVDO only, No 4G LTE support	USB
Pantech UMW190	V5.12	Verizon	USB

* - Currently Haute RouterOS works with PPP 3G modems over serial interfaces, 4G modems with IP drivers are not supported.

4G LTE cards

Model	Tested RouterOS version	Comments	Format
BandRich C501 [87]	v5.12	Configure under the new "/interface lte" menu	USB

Hardware reset of WRAPSXC and WRAPDXC routers

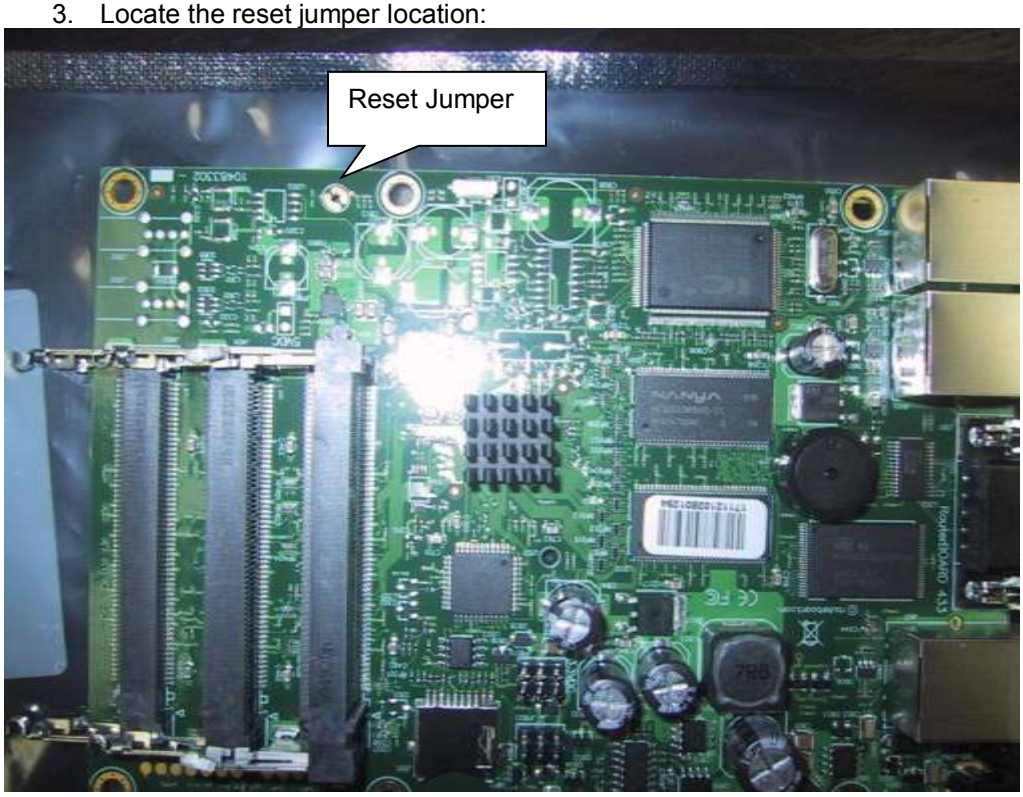
In the event that you have completely locked yourself out of your router, there is only one way to recover a router with a hardware reset.

For security and warranty reasons this procedure will void the warranty of a router and should only be executed after you have exhausted all other options to recover the unit through other means. Call HauteSpot Networks technical support before attempting to reset your unit using this proceedure.

Steps to recovery:

1. Remove power from the router by disconnecting the Power Over Ethernet connection to Ether1.
2. Remove the top cover of the router by removing the 4 top cover bolts using an 8mm socket wrench.

3. Locate the reset jumper location:



4. With a metal object such as a paperclip or screwdriver, short together the two sides of the reset jumper and hold while applying power by reconnecting the PoE cable to Ether1.
5. The router will now default to a completely blank configuration. It will not have an IP address and all configuration information will be lost.
6. You can log into the router using HSConfig and MAC address connections.
7. Once connected, you can restore the HauteSpot default configuration by running the import autorun.rsc command from a terminal window. This will reset the IP address to 192.168.1.1, set the router identity, create bridges, configure wireless interfaces, and in general bring the router to a usable state. Otherwise you can manually configure the router from scratch.

Overview of TDMA Like Protocol v2 protocol

TLPv2 protocol is proprietary wireless protocol developed by HauteSpot Networks for use with Atheros 802.11 wireless chips TLPv2 is based on TDMA (Time Division Multiple Access) media access technology instead of CSMA (Carrier Sense Multiple Access) media access technology used in regular 802.11 devices.

TDMA media access technology solves hidden node problem and improves media usage, thus improving throughput and latency, especially in PtMP networks.

Media access in TLPv2 network is controlled by TLPv2 Access Point. TLPv2 AP divides time in fixed size "periods" which are dynamically divided in downlink (data sent from AP to clients) and uplink (data sent from clients to AP) portions, based on queue state on AP and clients. Uplink time is further divided between connected clients based on their requirements for bandwidth. At

the beginning of each period AP broadcasts schedule that tells clients when they should transmit and the amount of time they can use.

In order to allow new clients to connect, TLPv2 AP periodically assigns uplink time for "unspecified" client - this time interval is then used by fresh client to initiate registration to AP. Then AP estimates propagation delay between AP and client and starts periodically scheduling uplink time for this client in order to complete registration and receive data from client.

TLPv2 implements dynamic rate selection on per-client basis and ARQ for data transmissions. This enables reliable communications across TLPv2 links.

For QoS TLPv2 implements variable number of priority queues with built-in default QoS scheduler that can be accompanied with fine grained QoS policy based on firewall rules or priority information propagated across network using VLAN priority or MPLS EXP bits.

Compatibility and coexistence with other wireless protocols

TLPv2 protocol is not compatible to or based on any other available wireless protocols or implementations, either TDMA based or any other kind, including Motorola Canopy, Ubiquiti Airmax and FreeBSD TDMA implementation. This implies that only TLPv2 supporting and enabled devices can participate in TLPv2 network.

Regular 802.11 devices will not recognize and will not be able to connect to TLPv2 AP. HauteRouterOS devices that have TLPv2 support (that is - have HauteRouterOS version 4.17 or higher) will see TLPv2 APs when issuing scan command, but will only connect to TLPv2 AP if properly configured.

As TLPv2 does not use CSMA technology it may disturb any other network in the same channel. In the same way other networks may disturb TLPv2 network, because every other signal is considered noise.

The key points regarding compatibility and coexistence:

- only HauteRouterOS devices will be able to participate in TLPv2 network
- only HauteRouterOS devices will see TLPv2 AP when scanning
- TLPv2 network will disturb other networks in the same channel
- TLPv2 network may be affected by any (TLPv2 or not) other networks in the same channel
- TLPv2 enabled device will not connect to any other TDMA based network

TLPv2 vs 802.11

The key differences between TLPv2 and 802.11:

Media access is scheduled by AP - this eliminates hidden node problem and allows you to implement centralized media access policy - AP controls how much time is used by every client and can assign time to clients according to some policy instead of every device contending for media access.

Reduced propagation delay overhead - There are no per-frame ACKs in TLPv2 - this significantly improves throughput, especially on long distance links where data frame and following ACK frame propagation delay significantly reduces the effectiveness of media usage.
Reduced per frame overhead - TLPv2 implements frame aggregation and fragmentation to maximize assigned media usage and reduce per-frame overhead (interframe spaces, preambles).

TLPv2 vs TLPv1

The key differences between TLPv2 and TLPv1:

Reduced polling overhead - instead of polling each client, TLPv2 AP broadcasts uplink schedule that assigns time to multiple clients, this can be considered "group polling" - no time is wasted for polling each client individually, leaving more time for actual data transmission. This improves throughput, especially in PtMP configurations.
Reduced propagation delay overhead - TLPv2 must not poll each client individually, this allows to create uplink schedule based on estimated distance (propagation delay) to clients such that media usage is most effective. This improves throughput, especially in PtMP configurations.
More control over latency - reduced overhead, adjustable period size and QoS features allows for more control over latency in the network.

Configuring TLPv2

As of version 4.17 new wireless interface setting wireless-protocol has been introduced. This setting controls which wireless protocol selects and uses. Note that meaning of this setting depends on interface role (either it is AP or client) that depends on interface mode setting. Find possible values of wireless-protocol and their meaning in table below.
Note that in the user interface TLPv1 is referred to as nstreme and TLPv2 is referred to as nv2. This is due the OEM nature of the interface versus the HauteSpot built driver. Wherever you see nv2 think TLPv2.

value	AP	client
unspecified	establish TLPv1 or 802.11 network based on old TLPv1 setting	connect to TLPv1 or 802.11 network based on old TLPv1 setting
any	same as unspecified	scan for all matching networks, no matter what protocol, connect using protocol of chosen network
802.11	establish 802.11 network	connect to 802.11 networks only
nstreme	establish TLPv1 network	connect to TLPv1 networks only
Nv2	establish Nv2 network	connect to Nv2 networks only
Nv2-nstreme-802.11	establish Nv2 network	scan for Nv2 networks, if suitable network found - connect, otherwise scan for TLPv1 networks, if suitable network found - connect, otherwise scan for 802.11 network and if suitable network found - connect.

Nv2-nstreme	establish Nv2 network	scan for Nv2 networks, if suitable network found - connect, otherwise scan for TLPv1 networks and if suitable network found - connect

Note that wireless-protocol values Nv2-nstreme-802.11 and Nv2-nstreme DO NOT specify some hybrid or special kind of protocol - these values are implemented to simplify client configuration when protocol of network that client must connect to can change. Using these values can help in migrating network to Nv2 protocol.

Most of Nv2 settings are significant only to Nv2 AP - Nv2 client automatically adapts necessary settings from AP. The following settings are relevant to Nv2 AP:

Nv2-queue-count - specifies how many priority queues are used in Nv2 network.
Nv2-qos - controls frame to priority queue mapping policy.
Nv2-cell-radius - specifies distance to farthest client in Nv2 network in km. This setting affects the size of contention time slot that AP allocates for clients to initiate connection and also size of time slots used for estimating distance to client. If this setting is too small, clients that are farther away may have trouble connecting and/or disconnect with "ranging timeout" error. Although during normal operation the effect of this setting should be negligible, in order to maintain maximum performance, it is advised to not increase this setting if not necessary, so AP is not reserving time that is actually never used, but instead allocates it for actual data transfer.
tdma-period-size - specifies size in ms of time periods that TLPv2 AP for media access scheduling. Smaller period can potentially decrease latency (because AP can assign time for client sooner), but will increase protocol overhead and therefore decrease throughput. On the other hand - increasing period will increase throughput but also increase latency. It may be required to increase this value for especially long links to get acceptable throughput. This necessity can be caused by the fact that there is "propagation gap" between downlink (from AP to clients) and uplink (from clients to AP) data during which no data transfer is happening. This gap is necessary because client must receive last frame from AP - this happens after propagation delay after AP's transmission, and only then client can transmit - as a result frame from client arrives at AP after propagation delay after client's transmission (so the gap is propagation delay times two). The longer the distance, the bigger is necessary propagation gap in every period. If propagation gap takes significant portion of period, actual throughput may become unacceptable and period size should get increased at the expense of increased latency. Basically value of this setting must be carefully chosen to maximize throughput but also to keep latency at acceptable levels.

The following settings are significant on both - TLPv2 AP and TLPv2 client:

- **Nv2-security** - specifies TLPv2 security mode
- **Nv2-preshared-key** - specifies preshared key to be used

Migrating to TLPv2

Using wireless-protocol setting aids in migration or evaluating TLPv2 protocol in existing networks really simple and reduce downtime as much as possible. These are the recommended steps:

- upgrade AP to version that supports TLPv2, but do not enable TLPv2 on AP yet.
- upgrade clients to version that supports TLPv2

- configure all clients with wireless-protocol=Nv2-nstreme-802.11. Clients will still connect to AP using protocol that was used previously, because AP is not changed over to TLPv2 yet
- configure TLPv2 related settings on AP
- if it is necessary to use data encryption and secure authentication, configure TLPv2 security related settings on AP and clients.
- set wireless-protocol=Nv2 on AP. This will make AP to change to TLPv2 protocol. Clients should now connect using TLPv2 protocol.
- in case of some trouble you can easily switch back to previous protocol by simply changing it back to whatever was used before on AP.
- fine tune TLPv2 related settings to get acceptable latency and throughput
- implement QoS policy for maximum performance.

The basic troubleshooting guide:

- clients have trouble connecting or disconnect with "ranging timeout" error - check that TLPv2-cell-radius setting is set appropriately
- unexpectedly low throughput on long distance links although signal and rate is fine - try to increase tdma-period-size setting

QoS in TLPv2 network

QoS in TLPv2 is implemented by means of variable number of priority queues. Queue is considered for transmission based on rule recommended by 802.1D-2004 - only if all higher priority queues are empty. In practice this means that at first all frames from queue with higher priority will be sent, and only then next queue is considered. Therefore QoS policy must be designed with care so that higher priority queues do not make lower priority queues starve.

QoS policy in TLPv2 network is controlled by AP, clients adapt policy from AP. On AP QoS policy is configured with Nv2-queue-count and Nv2-qos parameters. Nv2-queue-count parameter specifies number of priority queues used. Mapping of frames to queues is controlled by Nv2-qos parameter.
Nv2-qos=default

In this mode outgoing frame at first is inspected by built-in QoS policy algorithm that selects queue based on packet type and size. If built-in rules do not match, queue is selected based on frame priority field, as in Nv2-qos=frame-priority mode.
Nv2-qos=frame-priority

In this mode QoS queue is selected based on frame priority field. Note that frame priority field is not some field in headers and therefore it is valid only while packet is processed by given device. Frame priority field must be set either explicitly by firewall rules or implicitly from ingress priority by frame forwarding process, for example, from MPLS EXP bits.
Queue is selected based on frame priority according to 802.1D recommended user priority to traffic class mapping. Mapping depends on number of available queues (Nv2-queue-count parameter). For example, if number of queues is 4, mapping is as follows (pay attention how this mapping resembles mapping used by WMM):

- priority 0,3 -> queue 0
- priority 1,2 -> queue 1
- priority 4,5 -> queue 2

- priority 6,7 -> queue 3

If number of queues is 2 (default), mapping is as follows:

- priority 0,1,2,3 -> queue 0
- priority 4,5,6,7 -> queue 1

If number of queues is 8 (maximum possible), mapping is as follows:

- priority 0 -> queue 2
- priority 1 -> queue 0
- priority 2 -> queue 1
- priority 3 -> queue 3
- priority 4 -> queue 4
- priority 5 -> queue 5
- priority 6 -> queue 6
- priority 7 -> queue 7

For other mappings, discussion on rationale for these mappings and recommended practices please see 802.1D-2004.

Security in TLPv2 network

TLPv2 security implementation has the following features:

- hardware accelerated data encryption using AES-CCM with 128 bit keys;
- 4-way handshake for key management (similar to that of 802.11i);
- preshared key authentication method (similar to that of 802.11i);
- periodically updated group keys (used for broadcast and multicast data).

Being proprietary protocol TLPv2 does not use security mechanisms of 802.11, therefore security configuration is different. Interface using TLPv2 protocol ignores security-profile setting. Instead, security is configured by the following interface settings:

Nv2-security - this setting enables/disables use of security in TLPv2 network. Note that when security is enabled on AP, it will not accept clients with disabled security. In the same way clients with enabled security will not connect to unsecure APs.

Nv2-preshared-key - preshared key to use for authentication. Data encryption keys are derived from preshared key during 4-way handshake. Preshared key must be the same in order for 2 devices to establish connection. If preshared key will differ, connection will time out because remote party will not be able to correctly interpret key exchange messages.

Power Over Ethernet

WARNING: The HauteWRAP HR-WRAP series of routers are designed specifically for an input voltage of between 11VDC and 27VDC. Using a power supply with an output voltage in excess of 28VDC will damage your system. Such damage is not covered under your product warranty. Only use power supplies provided by HauteSpot Networks.

Power-over-Ethernet (PoE) or "Active Ethernet" eliminates the need to run 90-240 VAC power to Wireless Access Points and other devices on a wired LAN. Using Power-over-Ethernet system installers need to run only a single CAT5 or CAT6 Ethernet cable that carries both power and data to each device. This allows greater flexibility in the locating of AP's and network devices and significantly decreasing installation costs in many cases.

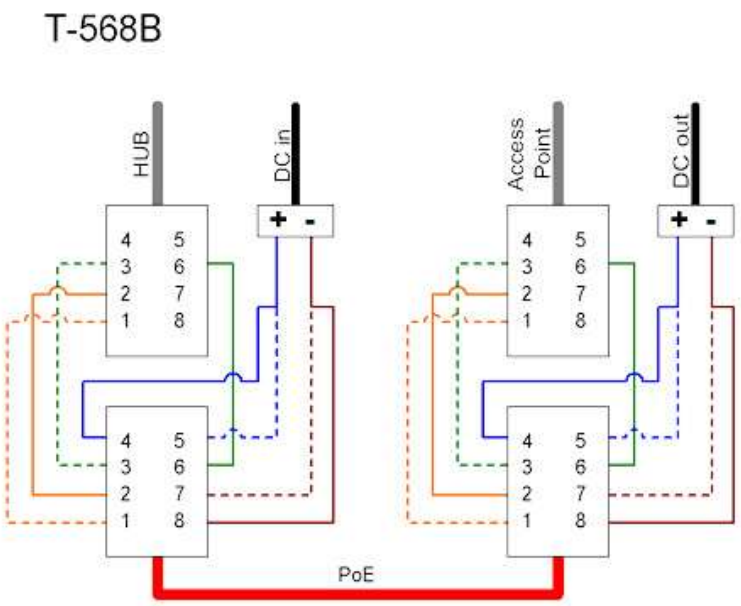
Power-over-Ethernet begins with a CAT5/6 "Injector" that inserts a DC Voltage onto the CAT5/6 cable. The Injector is typically installed in the "wiring closet" near the Ethernet switch or hub.

Our Wireless Access Points accept the injected DC power directly from the CAT5/6 cable through their NP1 (RJ45) jack. Our devices are considered to be "PoE-Compatible" or "Active Ethernet Compatible".

Voltage and Pinout Standards
Although the IEEE has a PoE standard called IEEE 802.3af, different equipment vendors use different PoE voltages and CAT5/6 pin configurations to provide the DC power. Therefore it is important to select the appropriate PoE devices for each piece of equipment you plan to power through the CAT5 cable.

The IEEE has standardized on the use of 48 VDC as the Injected PoE voltage. The HauteWRAP HR-WRAP series cannot accept this higher voltage. Where the maximum cable length has not been a major consideration the included 12 VDC is sufficient (up to 100 feet). For distances over 100 feet use the 18VDC adapter which can be purchased from HauteSpot Networks. **DO NOT USE other PoE solutions in combination with HauteSpot Networks devices.**

All of the HauteWRAP HR-WRAP products follow the IEEE standard for wiring. The following diagram shows how your cable from the injector to the HauteWRAP HR-WRAP unit should be wired. Included in your kit is a male RJ-45 connector and watertight fitting. This is provided so that you can measure the appropriate cable length in order to run up the mounting location of your system. Follow the attached diagram to create your lead wire and then crimp the connector on so that the water tight connector will seal the RJ-45 onto your unit. Power is delivered with positive voltage on pins 4/5 and negative voltage on pins 7/8. This is passive PoE and there is no negotiation. If using an active PoE hub or injector, you must disable active negotiation.



You should use a crimping tool to attach a male RJ-45 connector to your CAT 5/6 cable using the above diagram. This end will go to your POE injector. We recommend the use of 22AWG CAT6 cable if possible, as this will create less wire resistance, increasing the cable distance and improving reliability of your system.

You should use the following diagram to assemble the male end of your NP1 type waterproof connector to the other end of your CAT5/6 cable. This will attach to the HauteWRAP HR-WRAP unit.

Security on HauteSpot Networks HauteWRAP Wireless Routers

The HauteSpot Networks HauteWRAP product line, which includes the WRAPSTALC, WRAPSTA, WRAPLX, WRAPSXC, and WRAPDXC products, offers many advanced security features to authenticate users and devices and encrypt communications over both wired and wireless networks routed through our products.

Authentication is the process of identifying other devices or people and granting them appropriate permissions to access your device or network. HauteSpot products offer a variety of methods for device and user authentication.

The most basic method of authentication for people is *user id with password protection.* ***HauteSpot products all can be secured using simple password protection. Strong security will also encrypt connections into our devices using our HSConf tool.***

For devices, the most basic method of authentication is using *Access Control Lists (ACL)* based on MAC address. Every device with an Ethernet port has a unique MAC address. ***All HauteSpot products support filtering of access using lists to prevent unknown devices from connecting. These access control lists can be applied to any interface including wired, wireless and virtual***.

Certificated authentication uses a trusted third party, such as Verisign, CA-CERT, and Thawte, to validate the identity of users or devices. In using certificates, a user or device which wants to connect to another device first queries that device and is provided a public key, that public key, along with address and other information is then verified with the trusted third party to prove

identity. At the same time, the user or device trying to connect can provide the same information to the device being accessed. It too can validate the authenticity of the user requesting access. If the two devices or users are confirmed by the third party as being who they claim, then access can be granted.

All HauteSpot products support certification using standard protocols such as SSL, SSH and SCP. You can also create your own self signed certificates.

Another method of authentication is using Remote Authentication Dial In User Service or RADIUS. In RADIUS a database is run that stores user information. A router can then query that RADIUS server for user information and permissions. This method allows for dynamic keys that are provided to the user or device seeking access and the device granting access at the time of user authentication on the RADIUS server. Certificates can be used in conjunction with RADIUS to create even more secure connections.

All HauteSpot products support RADIUS EAP user authentication.

Once a user is authenticated, then encryption can be used to secure the communications between the user or device and the host device.

Encryption is the scrambling of data being exchanged between two devices using a necessarily complex algorithm (cipher) which starts with a common key. This key allows the encryption to be processed by only devices that hold this key. Typically once a user is authenticated, a private complex key is exchanged between the host device and the remote device. This key typically is changed every few seconds or minutes so that it is more difficult for an intruder to listen to the conversation and guess what the key is by looking for common repeated strings.

The algorithms or ciphers most typically used today for wireless encryption are the Advanced Encryption Standard or AES and Temporal Key Integrity Protocol or TKIP. HauteSpot products support both AES and TKIP, although we recommend securing your devices using AES only due to security flaws in the TKIP algorithm.

There are other ciphers that can be used to create secure tunnels that can run over wireless or wired interfaces such as Data Encryption Standard (DES), Triple Data Encryption Algorithm (3DES), and others. These algorithms can be used in tunnel schemes such as OpenVPN, PPTP, L2TP, and IPSec. These tunnels allow communication over any public or private network to assure complete security.

All HauteSpot products support a broad variety of tunneling and VPN protocols.

Wi-Fi Protected Access (**WPA** and **WPA2**) is a proprietary certification program developed by the Wi-Fi Alliance, a group of consumer WiFi equipment manufacturers, to indicate compliance with the proprietary security protocol they defined to secure wireless computer networks. The Alliance defined the protocol in response to several serious weaknesses researchers had found in the previous system, WEP (Wired Equivalent Privacy).

The WPA protocol implements most, but not all, of the IEEE 802.11i standard. The Wi-Fi Alliance rushed WPA to market in order to fill gaps identified in WEP, while the IEEE worked on finalizing 802.11i. Specifically the Wi-Fi Alliance implemented TKIP as the main algorithm in WPA. TKIP was implemented on pre-WPA wireless network interface cards sold by the Wi-Fi Alliance members as far back as 1999, but no standard for interoperability existed. Because the changes required fewer modifications on the client than on the wireless access points (APs), most pre-2003 APs could not be upgraded to support WPA with TKIP. Researchers have since discovered a flaw in TKIP that relied on older weaknesses to retrieve the keystream from short packets to use for re-injection and spoofing.

The later WPA2 certification mark indicates compliance with most of the IEEE 802.11i standard. This advanced protocol will not work with some older network cards. WPA2 has replaced WPA. WPA2 implements only the mandatory elements of 802.11i. In particular, it introduces a new AES-based algorithm, CCMP, which is considered fully secure.

All HauteSpot products implement the full IEEE 802.11i specification, of which WPA2 is a subset.

Pre-shared key mode (PSK, also known as *Personal* mode) is designed for home and small office networks that don't require the complexity of an 802.1X (RADIUS) authentication server. Each wireless network device encrypts the network traffic using a 256 bit key. According to 802.11i, this key may be entered either as a string of 64 hexadecimal digits. WPA2 also allows for entry of a passphrase of 8 to 63 printable ASCII characters. If ASCII characters are used, the 256 bit key is calculated by applying the PBKDF2 key derivation function to the passphrase, using the SSID as the salt and 4096 iterations of HMAC-SHA1. ***HauteSpot products do not implement the passphrase function, as this is not part of the 802.11i standard. If you are using a device that only supports passphrases, then HauteSpot has a calculator which will create the key based on the passphrase which can be entered into our routers.***

Shared-key WPA remains vulnerable to password cracking attacks if users rely on a weak passphrase. To protect against a brute force attack, a truly random passphrase of 13 characters (selected from the set of 95 permitted characters) is probably sufficient. Lookup tables have been computed by the Church of WiFi (a wireless security research group) for the top 1000 SSIDs for a million different WPA/WPA2 passphrases. To further protect against intrusion the network's SSID should not match any entry in the top 1000 SSIDs. Better yet, use a complex SSID and then hide it.

The Wi-Fi alliance has announced the inclusion of additional EAP (Extensible Authentication Protocol/RADIUS) types to its certification programs for WPA- and WPA2- Enterprise certification programs. This was to ensure that WPA-Enterprise certified products can interoperate with one another. Previously, only EAP-TLS (Transport Layer Security) was certified by the Wi-Fi alliance.

As of 2010 the certification program includes the following EAP types:

- EAP-TLS (previously tested)
- EAP-TTLS/MSCHAPv2
- PEAPv0/EAP-MSCHAPv2
- PEAPv1/EAP-GTC
- PEAP-TLS
- EAP-SIM

802.1X clients and servers developed by specific firms may support other EAP types. This certification is an attempt for popular EAP types to interoperate; their failure to do so is currently one of the major issues preventing rollout of 802.1X on heterogeneous networks.

HauteSpot products fully support EAP standard authentication using RADIUS as defined by the IEEE. HauteSpot also offers a free User Manager program which is embedded on every router that can store user information and be used as a RADIUS repository for authentication on the network. Contact HauteSpot Networks for information on using User Manager.

Additionally, the HauteSpot TDMA Like Protocol (TLP) is designed for enhanced security. It is not visible to any devices other than HauteSpot routers, yet is aware of 802.11 networks operating in the same area and avoids collisions and conflicts. By using TLP, you reduce the chance of someone even having the right equipment to hack your network.

Federal Information Processing Standards Publication (FIPS PUB) 140-2, *Security Requirements for Cryptographic Modules*, specifies the security requirements that are to be satisfied by the cryptographic module utilized within a security system protecting sensitive

information within computer and telecommunications systems (including voice systems). The standard provides four increasing, qualitative levels of security: Level 1, Level 2, Level 3, and Level 4. These levels are intended to cover the wide range of potential applications and environments in which cryptographic modules may be employed. The security requirements cover eleven areas related to the secure design and implementation of the cryptographic module. Both AES and 3DES encryption are specified in FIPS-140-2 as acceptable cryptographic modules. While HauteSpot has not tested it's products for FIPS compliance, we are confident that we meet the requirements for compliance if it were required.
Finally, through route filters, bridge filters and other methods, it is possible to limit access to the network or to specific devices on the network based on protocols, source addresses, destination addresses, data type, MAC address, time of day, or other such factors.
In summary, you can fully secure your HauteSpot Networks wireless router by doing the following:

1. Change your admin user password to use a complex password.
2. Use a complex SSID or hid the SSID.
3. If on a private network with known clients, use MAC Access Control Lists.
4. Use the HauteSpot TLP protocol.
5. Use AES encryption with dynamic keys (WPA2 ENTERPRISE) if possible. This is most secure. Or at minimum use AES with a pre-shared complex key (WPA2 PERSONAL).
6. Use HauteSpot User Manager to manage user accounts.
7. Use a VPN tunnel over the wireless network using an encryption algorithm for even more security.
8. Disable all unnecessary packages on your HauteSpot router. (/system packages)
9. Disable all unnecessary TCP ports on your HauteSpot router. (/ip services)
10. Limit address ranges from which administration of the router can occur. (/ip services)
11. Rotate passwords monthly.
12. Enable system security logging (/system logging) and review logs periodically.

Serial Port Usage

Usage possibilities for serial ports in HauteRouterOS

The most obvious (and default) usage is the serial console for HauteRouterOS. On every HR-WRAP or other system you will have a HauteRouterOS login prompt on serial0 (the first serial interface). This is one of the ways to do the basic configuration on your HauteRouterOS device after installation (i.e. at least give it an ip address or so).

There are other and more advanced uses for serial ports. I try to sum up those possibilities here.

You can use HauteRouterOS to access another device (like a switch, for example) that is connected through a serial cable

- directly from a HauteRouterOS console or telnet/ssh login (i.e. "from within HauteRouterOS")
- by opening a telnet/ssh session that will get you directly on this device (without having to login to HauteRouterOS first)
- as if it were connected to a physical COM port of your PC

Preparations

Please note that there are some Caveats you should be aware of! Take your time understanding those limits to avoid strange things to happen when connecting a device to a serial port on a HR-WRAP.

First you have to have a free serial port to use for this. If you are on a device with only one serial port (like all WRAPSXCs or WRAPDXCs.) you will have to disable the system console on this serial port. By default on “s” models this is done for you:

```
/system console disable 0
```

Be sure to just disable the console rather than removing it, as HauteRouterOS will recreate the console after the next reboot when you really remove it.

Next you will have to configure your serial port according to the serial port settings of the connected device. Using the following command you will set your serial port to 19200 Baud 8N1. What settings you need to use depends on the device you connect:

```
/port set serial0 baud-rate=19200 data-bits=8 parity=none stop-bits=1
```

You can also try to let HauteRouterOS guess the needed baud rate by setting

```
/port set serial0 baud-rate=auto
```

Now's the time to connect your device if not already done. Usually you will have to use a null modem cable (the same thing as a cross-over-cable for Ethernet). Now we're ready to go!

Accessing a serial device from within HauteRouterOS

The easiest way is the serial-terminal command:

```
/system serial-terminal serial0
```

This will give you access to the device you connected to port serial0. *Ctrl-A* is the prefix key, which means that you will enter a small "menu" by pressing *Ctrl-A*. The *Ctrl-A* character will not be sent to your device!

If you want to exit the connection to the serial device type *Ctrl-A*, then *Q*. This will return you to your HauteRouterOS console.

If you really want to send a *Ctrl-A* character to your serial device you have to type *Ctrl-A* two times.

Accessing a serial device directly by SSH / telnet / mac-telnet

If you want to access the device directly without logging into HauteRouterOS first, use the special-login command.

First create a new user for access to the serial device:

```
/user add name=serial password=secret group=read
```

Then connect this username to the desired serial port:

```
/special-login add user=serial port=serial0
```

Now when you connect to your HR-WRAP using SSH (or telnet/mac-telnet) and use the username/password of the user you just created, you will be connected to your serial device directly.

One nice side effect of this is that you could give this login to someone else who can access the serial device, but not HauteRouterOS itself. (Or, if connecting by HSConfig, couldn't change anything.)

Accessing a serial device as if it were physically connected to your PCs' COM port

Perhaps you are using software to access your serial device that can only talk to COM ports like COM1?

Want to use this to access your device when it's located somewhere else?

The usual way would be to purchase something like a LanTronix or Moxa serial server, but you can do this for free with HauteRouterOS, too!

RFC 2217 defines a protocol to transfer data from/to a serial device over TCP. Combine this with a software driver on your PC that presents this network connection as a "virtual COM port" and you're done!

There are lots of software solutions for the PC side to do this, most of them commercial. One that works well with HauteRouterOS in my experience is "Serial/IP COM Port Redirector" from Tactical Software (and no, I'm not affiliated with them anyhow ;-)).

Serial to Ethernet Connector lets you share up to 255 serial port devices over network (Internet or LAN).

But there's also a free solution that you may even use commercially! It's named VSPE (virtual serial port emulator) and can be obtained from Eterlogic.

Also could be useful freeware RS232 Data Logger by Eltima

Configuring the HS-WRAP

```
/port remote-access add port=serial0 protocol=rfc2217 tcp-port=10000
```

This will connect an incoming connection on TCP port 10000 to the port serial0.

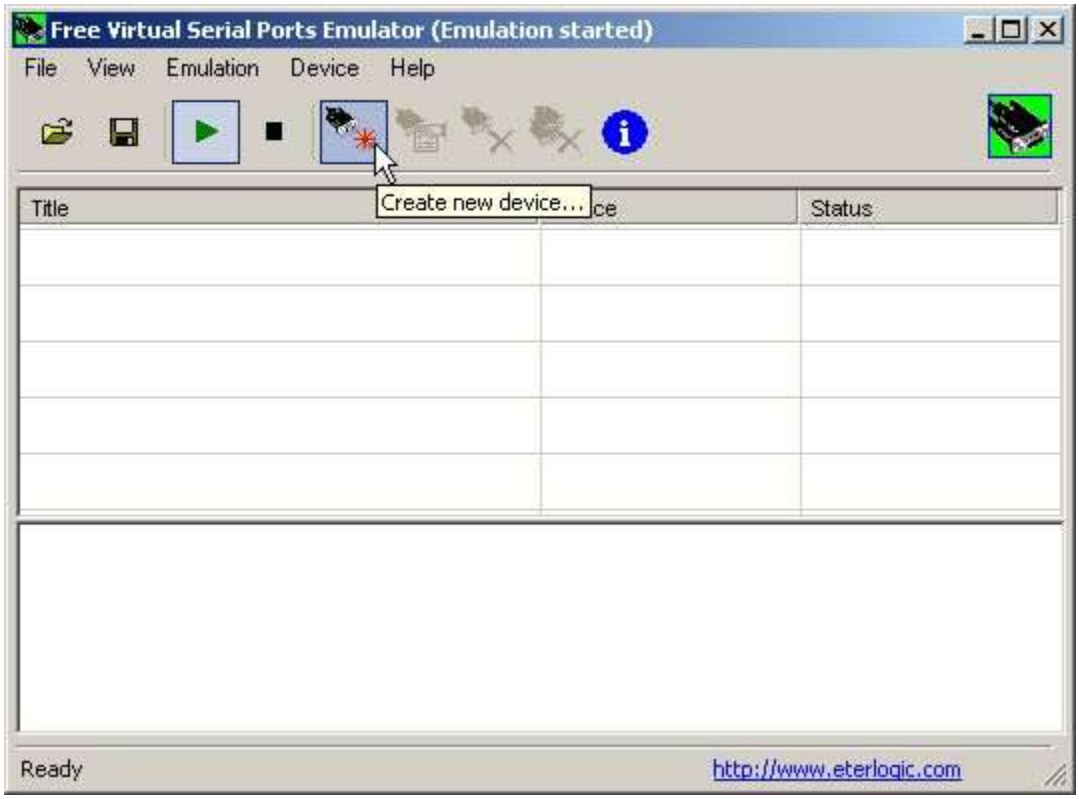
You can restrict from what ip addresses you accept connections using the parameter address-range (as in "/ip service"). Using the "local-address" parameter you can specify on which local ip address you want to listen for incoming connections (if you have more than one ip address configured).

That's about everything we need to configure in HauteRouterOS!

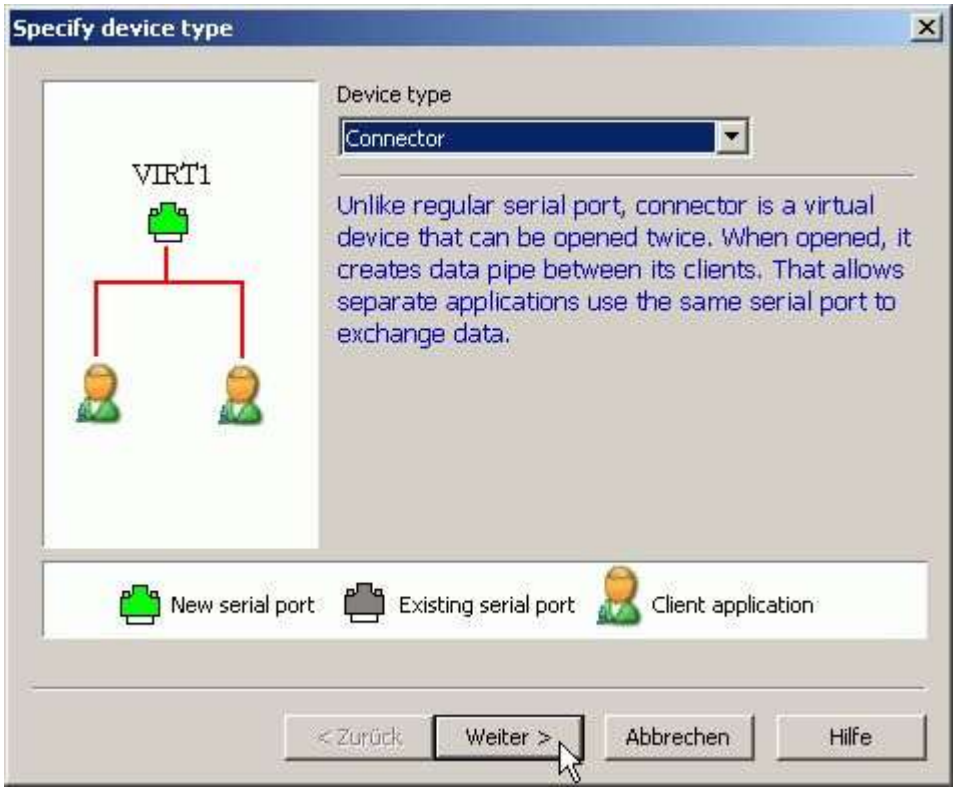
Configuring your PC

Now on to the PC part. Install the VPSE software from Eterlogic and start it.

You will see a screen like this:



Clicking on the "Create new device" button will get you here:

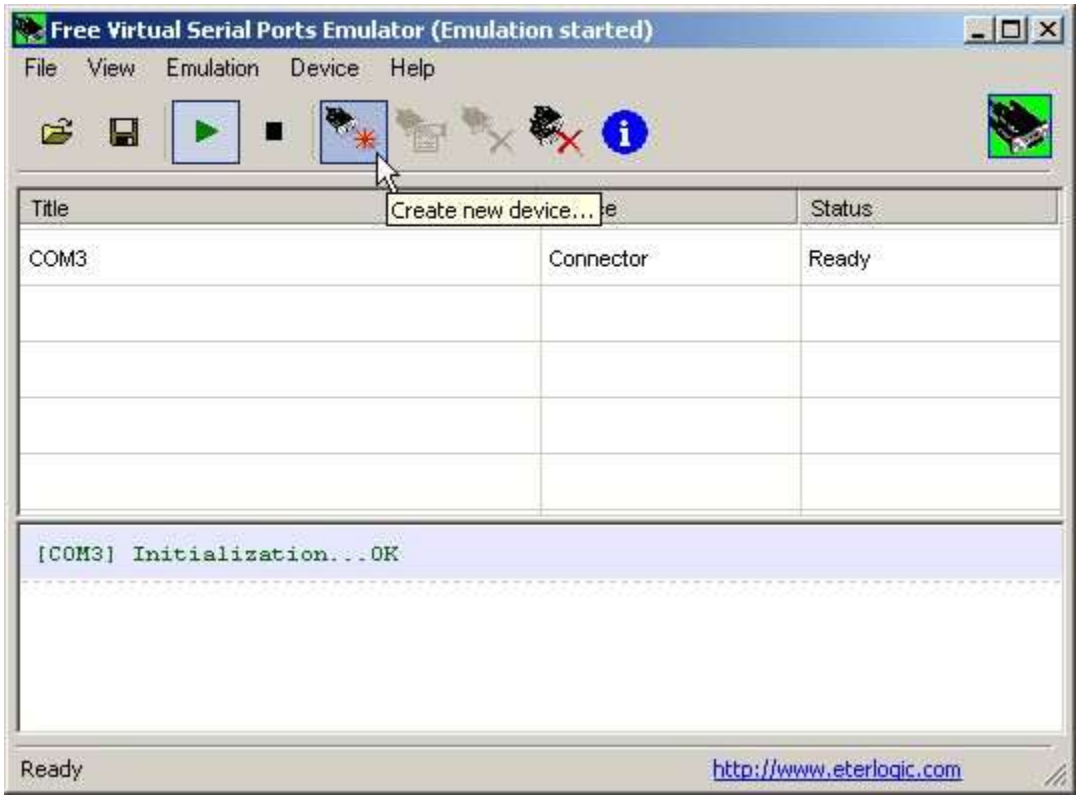

Specify device type
VIRT1
Device type
Connector
Unlike regular serial port, connector is a virtual device that can be opened twice. When opened, it creates data pipe between its clients. That allows separate applications use the same serial port to exchange data.
New serial port
Existing serial port
Client application
< Zurück
Weiter >
Abbrechen
Hilfe

Make sure you selected device type "Connector" in the drop-down and press "Next":

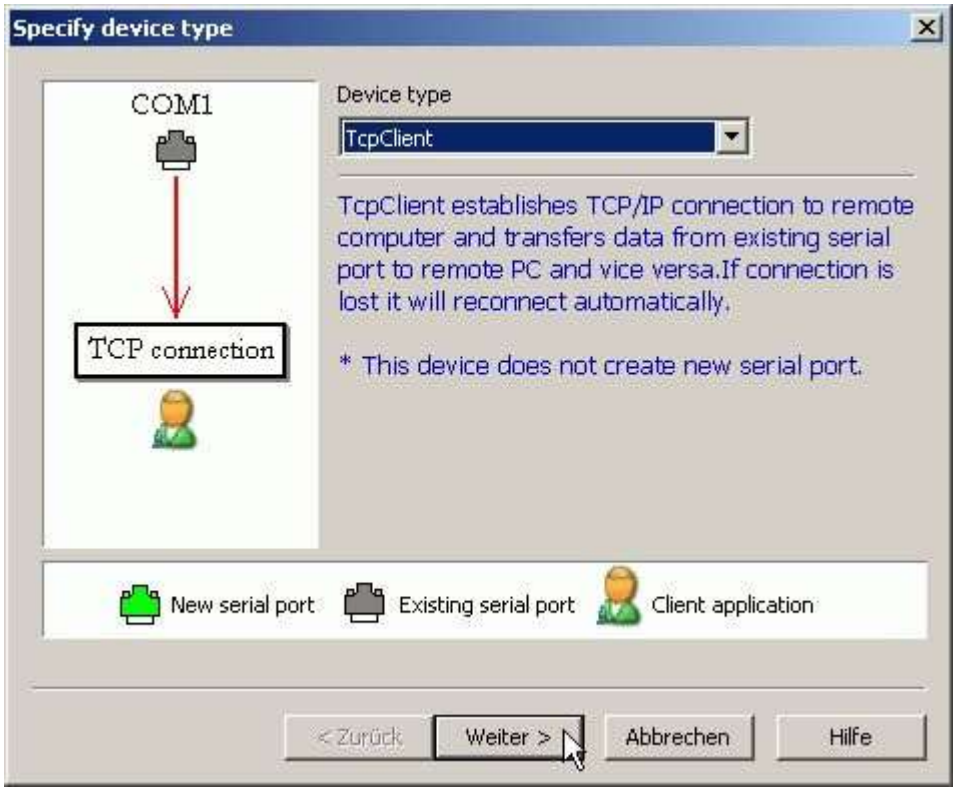
Now select a (unused) COM port number which should be your new virtual COM port and press "Finish".

Make sure that the application you want to use to access this virtual port can open the COM port number you are about to create here. Older applications often only can be configured to use COM1-COM4, so creating COM9 could mean getting yourself into trouble.

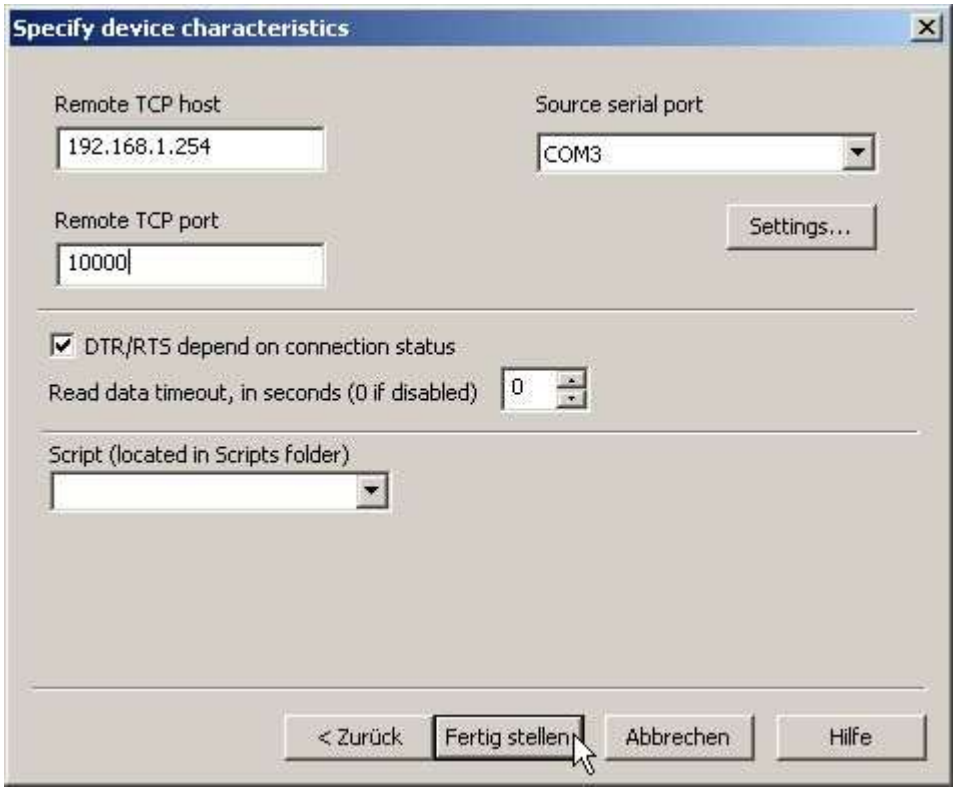
Verify that the virtual COM port you created (COM3 in the screenshot) is shown as "Ready":



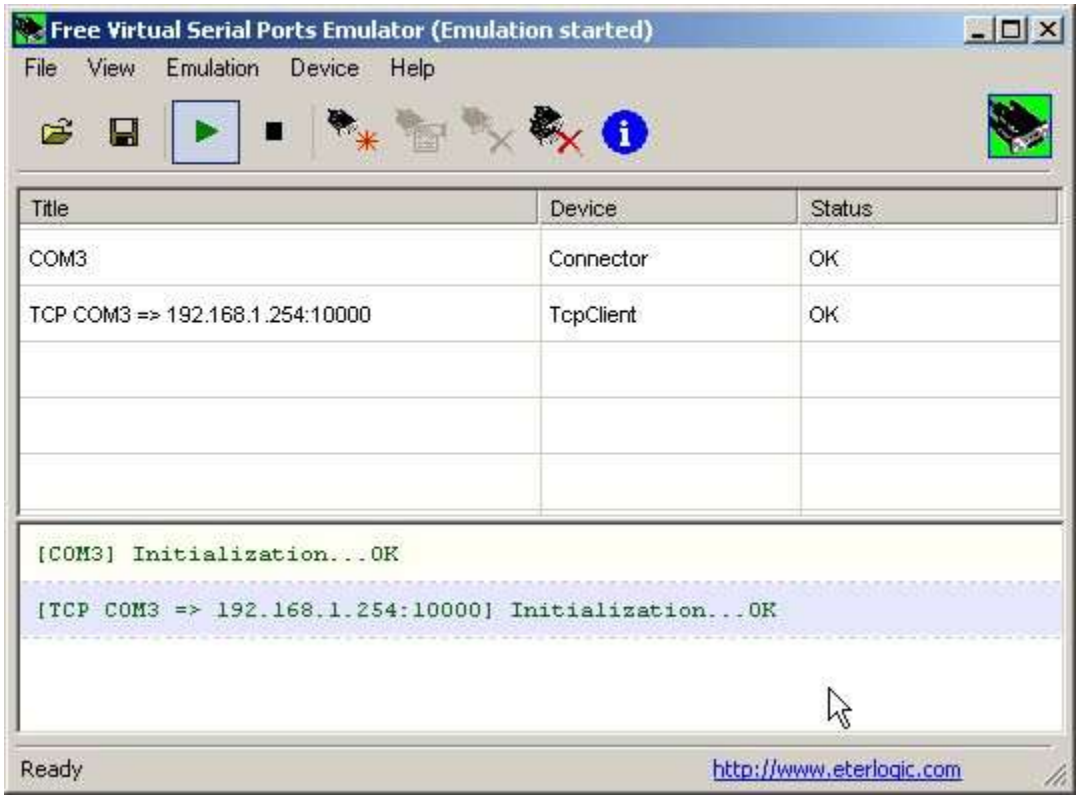
Next, press the "Create new device" button again and select "TcpClient" as device type:


Specify device type
COM1
TCP connection
Device type
TcpClient
TcpClient establishes TCP/IP connection to remote computer and transfers data from existing serial port to remote PC and vice versa.If connection is lost it will reconnect automatically.
* This device does not create new serial port.
New serial port
Existing serial port
Client application
< Zurück
Weiter >
Abbrechen
Hilfe

Press "Next". In the next screen you will have to enter the ip address of your HR-WRAP (192.168.1.254 in the screenshot) and the port number you configured above (10000 in our example). Also make sure that you select the virtual COM port we created just before in the "Source serial port" drop-down:



Clicking on "Finish" will end our work and present you with a screen like this:



If the status of your TCPClient device doesn't show "OK", you either have an error in your HauteRouterOS config, or you cannot reach the configured port on the HauteRouterOS machine from your PC (network cabling? routing? firewall? ...).

If everything went well, you can now start any software and use our freshly created COM3 (in the example) to access your serial device connected to the HauteRouterOS machine. Even if this would be located on another continent!

Caveats

There are some caveats with using the serial port on HR-WRAPs for the applications described above. Please read them carefully to understand how you might lock yourself out of your router!

Losing serial console access

By reconfiguring port serial0 on a HR-WRAP as seen above, you will lose your serial console access to HauteRouterOS. This means, that if you cannot access your HR-WRAP over the network anymore, you might even have to reset the whole configuration of it to gain access again.

HauteRouterBOOT entering the game

When rebooting a HR-WRAP the bootloader (HauteRouterBOOT) will always use the serial console (serial0 on HR-WRAPs) to send out some startup messages and offer access to the HauteRouterBOOT menu.

Having text coming out of the serial port to the connected device might confuse your attached device. Furthermore in standard config you can enter HauteRouterBOOT menu by pressing ANY key. So if your serial device sends any character to the serial port of your HR-WRAP during boot time, the HR-WRAP will enter the HauteRouterBOOT menu and will NOT boot HauteRouterOS unless you manually intervene!

You can reconfigure HauteRouterBOOT to enter the HauteRouterBOOT menu only when a DEL character is received - use this to reduce the chance to get a router that's stuck when rebooting!

Or in newer versions (4.11 or higher) are used "Silent boot" feature can be used to suppress any output on the serial interface, including remove booting sounds.

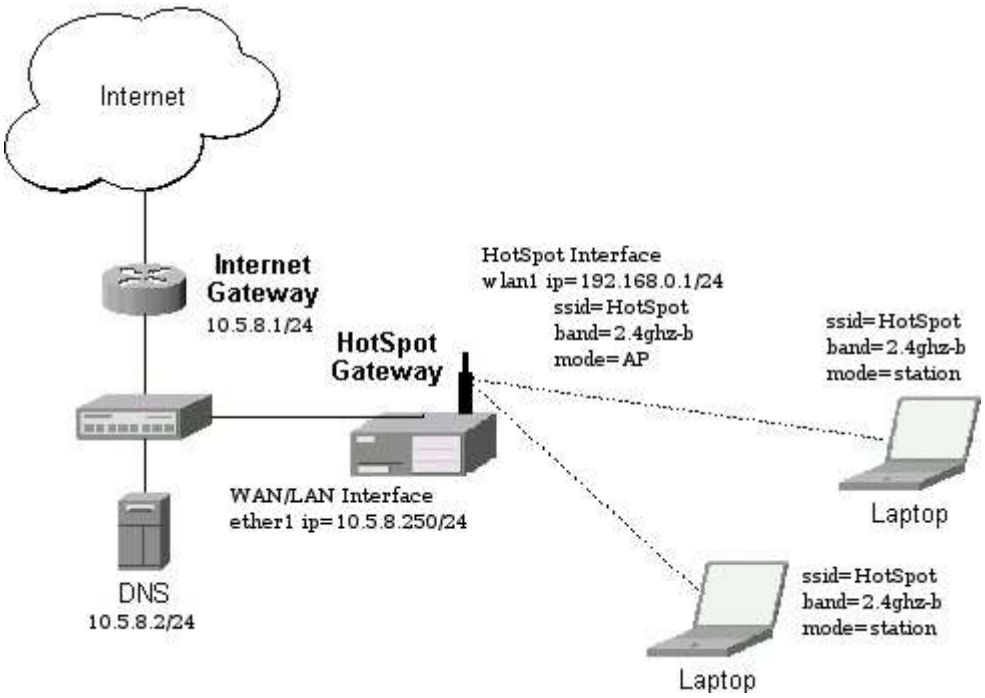
Simple Guide to make a HotSpot gateway

The HauteSpot HotSpot Gateway enables providing of public network access for clients using wireless or wired network connections. HotSpot Gateway should have at least two network interfaces:

1. HotSpot interface, which is used to connect HotSpot clients

2. LAN/WAN interface, which is used to access network resources.

The following picture shows wireless HotSpot setup



To setup simple HotSpot Gateway follow the steps below:

1. Configure wireless interface on HotSpot Gateway:

```
[admin@HotSpot_Gateway]> interface wireless set wlan1 ssid=HotSpot
band=2.4ghz-b \
\... mode=ap-bridge
```

2. Configure ip address for HotSpot interface:

```
[admin@HotSpot_Gateway] > ip add add address=192.168.0.1/24
interface=wlan1
```

3. Configure ip address for WAN/LAN interface:

```
[admin@HotSpot_Gateway] > ip add add address=10.5.8.250/24
interface=ether1
```

4. Add a route on HotSpot Gateway

```
[admin@HotSpot_Gateway] > ip route add gateway=10.5.8.1
```

5. Configure Hotspot on wlan1 interface and add user admin with pasword test

```
[admin@HauteSpot] > ip hotspot setup
hotspot interface: wlan1
```

```
local address of network: 192.168.0.1/24
masquerade network: yes
address pool of network: 192.168.0.2-192.168.0.254
select certificate: none
ip address of smtp server: 0.0.0.0
dns servers: 10.5.8.2
dns name: hs.hautespot.net
name of local hotspot user: admin
password for the user: test
```

In order to access network resources HotSpot clients have to configure their wireless interfaces setting proper ssid, band and mode, and enabling dynamic host configuration (dhcp) on the wireless interface.

Next step is you need to change login page here is one example

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html>
<head>
        <title>HauteSpot hotspot > login</title>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8" />
        <meta http-equiv="pragma" content="no-cache" />
        <meta http-equiv="expires" content="-1" />
        <style type="text/css">
        body {font-family: verdana;
                        text-align: center;
                        min-width:800px;
                        background-color: #ffffff;
                        float:center;
                        font-family:Palatino Linotype,Tahoma,Lucida
Grande,sans-serif;
                        }
        img {border: none;}
        a, a:link, a:visited, a:active { text-decoration: none; }
        #all { display:table; width:800px; margin: 0px auto;}
        #wrapper { border:1px solid #c1c1cc;  clear:both; }
        #header { font-size: 50px; color: #006FAA; font-style: italic;
font-weight: bold;}
        #left { width:60%; font-size: 25px; color: #27764A; float:left;
margin : 30px 5px 5px 10px;  border-top :1px solid #c1c1cc; padding-
top: 20px;}
        #click { font-size: 20px; color: #131EB5;  }
        #center { font-size: 20px; color: #131EB5; text-align: left; }
        #right  { width:30%; margin-top: 30px; margin-left: 15px;
float:left; padding-top: 20px; font-size: 20px; border: none;}
        #adv   {margin-top: 10px;}
        #footer {color: #c1c1c1; font-size: 9px; margin-top: 10px;
clear:both; font-family: verdana;}
        </style>

</head>
        <body>
```

```
        <div id="all">
        <div id="wrapper">
        <div id="header"><img src="img/logo.jpeg"> Free internet
access</div>


        <div id="left">
        This is free internet acceess with advertizement.
        <br>Add you text here
        <br><br>To process using please click below
        <br> $(if trial == 'yes')<a id="click" href="$(link-login-
only)?dst=$(link-orig-esc)&username=T-$(mac-esc)">click
here</a>.$(endif)
        $(if trial != 'yes')Access is temporarily disabled .$(endif)

        <div id="center">
        <br>Also you can see our sponsor page:
        <br> page from wall garden
        <div style="color: #FF8080; font-size: 15px; margin-left: 25px;
text-decoration: none; ">
        <br><a href="http://www.HauteSpot.net"> HauteSpot </a>
        <br><a href="http://www.HauteSpot.net"> HauteSpot 2</a>
        </div>
        </div>
        </div>
        <div id="right">
        <b>Sponsor by:</b>
        <br>
    <a   href="http://www.HauteSpot.net"> <img id="adv"
src="img/logo.jpeg"> </a><br>
    <a   href="http://www.HauteSpot.net"> <img id="adv"
src="img/Sponsor2.jpeg"> </a><br>
    <a   href="http://www.HauteSpot.net"> <img id="adv"
src="img/Sponsor3.jpeg"> </a><br>
    <a   href="http://www.HauteSpot.net"> <img id="adv"
src="img/Sponsor4.jpeg"> </a><br>
        </div>


        <div id="footer">Powered by HauteSpot Networks © 2010
HauteSpot</div>
        $(if error)<br /><div style="color: #FF8080; font-size:
9px">$(error)
        </div>$(endif)


        </div>

        </div>
  </body>
</html>
```

```
<div id="header"><img src="img/logo.jpeg"> Free internet access</div>
```

This line define logo picture in directory img you can replace with you own logo.

```
<div id="left">
        This is free internet acceess with advertizement.
        <br>Add you text here
        <br><br>To process using please click below
        <br> $(if trial == 'yes')<a id="click" href="$(link-login-
only)?dst=$(link-orig-esc)&username=T-$(mac-esc)">click
here</a>.$(endif)
        $(if trial != 'yes')Access is temporarily disabled .$(endif)
```

In this section you can define custom text for page and also have define where user to click to process using internet. If trial user account disable you can also define text.

```
<br>Also you can see our sponsor page:
        <br> page from wall garden
        <div style="color: #FF8080; font-size: 15px; margin-left: 25px;
text-decoration: none; ">
        <br><a href="http://www.HauteSpot.net"> HauteSpot </a>
        <br><a href="http://www.HauteSpot.net"> HauteSpot 2</a>
```

This section login page is for pages garden in wall garden user without clicking can access, this can be your web page or page some sponsors.
You can add some custom text and links to web page like in example.

```
<b>Sponsor by:</b>
        <br>
    <a   href="http://www.HauteSpot.net"> <img id="adv"
src="img/logo.jpeg"> </a><br>
    <a   href="http://www.HauteSpot.net"> <img id="adv"
src="img/Sponsor2.jpeg"> </a><br>
    <a   href="http://www.HauteSpot.net"> <img id="adv"
src="img/Sponsor3.jpeg"> </a><br>
    <a   href="http://www.HauteSpot.net”> <img id="adv"
src="img/Sponsor4.jpeg"> </a><br>
        </div>
```

Now you can define small banners for you sponsors or something like this. Also when user click on in open his web page, also you can add this page in wall garden to allow user without login to browsing this page.

Transparently Bridge two Networks using MPLS

Overview

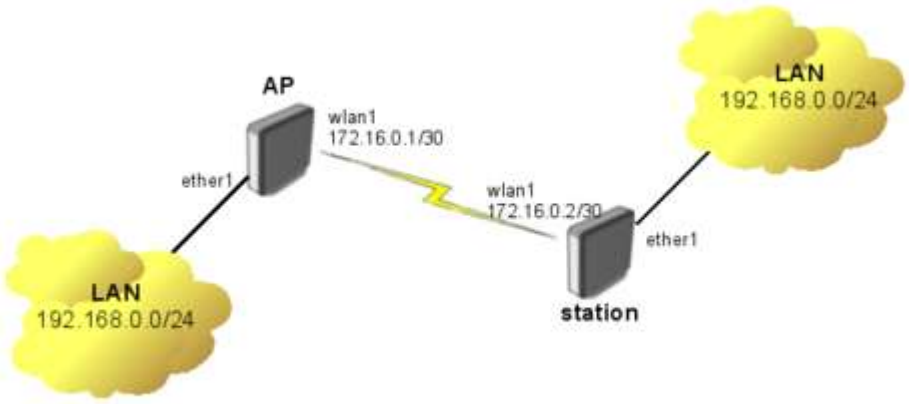
This a very basic example how to enable MPLS, establish VPLS tunnel between two wireless links and use it to transparently bridge two networks.

The MPLS/VPLS approach has some advantages over WDS or EoIP:

- VPLS tunnel is about 60% faster and less overhead than EoIP tunnel
- 802.11n speed is limited over WDS bridges, this method doesn't have such limitations

Configuration

Let us assume the following network setup:



AP

```
# --configure wireless access point--
/interface wireless
set wlan1 disabled=no ssid=MPLS frequency=5180 band=5ghz mode=bridge

# --configure IP--
/ip address
add address=172.16.0.1/30 interface=wlan1

# --enable LDP--
/mpls ldp
set enabled=yes lsr-id=172.16.0.1 transport-address=172.16.0.1
/mpls ldp interface
```

```
add interface=wlan1

# --configure VPLS tunnel--
/interface vpls
add name=vpls1 remote-peer=172.16.0.2 vpls-id=1:1 disabled=no

# --add bridge and bridge ports --
/interface bridge add name=bridge1
/interface bridge port
add bridge=bridge1 interface=ether1
add bridge=bridge1 interface=vpls1
```

station

```
# --configure wireless access point--
/interface wireless
set wlan1 disabled=no ssid=MPLS band=5ghz mode=station

# --configure IP--
/ip address
add address=172.16.0.2/30 interface=wlan1

# --enable LDP--
/mpls ldp
set enabled=yes lsr-id=172.16.0.2 transport-address=172.16.0.2
/mpls ldp interface
add interface=wlan1

# --configure VPLS tunnel--
/interface vpls
add name=vpls1 remote-peer=172.16.0.1 vpls-id=1:1 disabled=no

# --add bridge and bridge ports --
/interface bridge add name=bridge1
/interface bridge port
add bridge=bridge1 interface=ether1
add bridge=bridge1 interface=vpls1
```

Confirm that LDP is running

```
[admin@HauteSpot] /mpls ldp neighbor> print
Flags: X - disabled, D - dynamic, O - operational, T - sending-targeted-hello, V - vpls
 #      TRANSPORT    LOCAL-TRANSPORT PEER           SEND-TARGETED ADDRESSES
 0 DOTV 172.16.0.2   172.16.0.1      172.16.0.2:0   no            172.16.0.2

[admin@HauteSpot] /mpls ldp neighbor> .. .. forwarding-table print
Flags: L - ldp, V - vpls, T - traffic-eng
 #   IN-LABEL        OUT-LABELS      DESTINATION    INTERFACE    NEXTHOP
 0   expl-null
 1 V 18                              vpls1
```

Confirm that VPLS tunnel is established:

```
[admin@HauteSpot] /interface vpls> monitor vpls1 once
       remote-label: 17
        local-label: 18
      remote-status:
  transport-nexthop: 172.16.0.2
     imposed-labels: 17
```

Connect more stations to the same Bridge

It is possibile to connect more than one Station to the same Bridge, simply creating more VPLS tunnels (one for Station):

AP

```
# --change wireless access point mode to AP Bridge--
/interface wireless
set wlan1 mode=ap-bridge

# --configure second VPLS tunnel--
/interface vpls
add name=vpls2 remote-peer=172.16.0.3 vpls-id=2:2 disabled=no

# --add vpls2 to bridge --

/interface bridge port
add bridge=bridge1 interface=vpls2
```

2nd Station

```
# --configure wireless access point--
/interface wireless
set wlan1 disabled=no ssid=MPLS band=5ghz mode=station

# --configure IP--
/ip address
add address=172.16.0.3/30 interface=wlan1

# --enable LDP--
/mpls ldp
set enabled=yes lsr-id=172.16.0.3 transport-address=172.16.0.3
/mpls ldp interface
add interface=wlan1

# --configure VPLS tunnel--
/interface vpls
add name=vpls1 remote-peer=172.16.0.1 vpls-id=2:2 disabled=no

# --add bridge and bridge ports --
/interface bridge add name=bridge1
/interface bridge port
add bridge=bridge1 interface=ether1
add bridge=bridge1 interface=vpls1
```

Note

When router encapsulates Ethernet frame to forward over VPLS pseudowire, it checks if packet size + VPLS CW + MPLS labels exceeds MPLS MTU of outgoing interface. If it does, VPLS will fragment frame. In this example 1514byte layer2 packets are forwarded over VPLS, router adds CW (8bytes) and one MPLS tag (4bytes) it means that to avoid fragmentation MPLS MTU must be increased to 1526

```
/mpls interface set 0 mpls-mtu=1526
```

Ethernet port should be capable of sending 1526 byte L2MTU packets.

Note: If interface does not support L2MTU specified as mpls-mtu, then packets will be silently dropped

Custom Radio Module Options for HauteSpot WRAP Routers

Military/Federal Bands

Frequency Range	Availability
200 - 400 MHz VHF band	Now
375 - 420 MHz	Now
440 - 510 MHz	Now
902 - 928 MHz	Now
960 - 1170 MHz	Now
1160 - 1240 MHz	Now
1300 - 1390 MHz	Now
1435 - 1525 MHz	Now
2335 - 2535 MHz	Now
4400 - 5000 MHz	Now
4800 - 5000 MHz	Now
5000 - 5150 MHz	Now

Public Safety Bands

Frequency Range	Availability
450 - 512 MHz	Now
756 - 784 MHz	Now
788 - 799 MHz	Now
4940 - 4990 MHz	Now

Broadband Infrastructure Bands

Frequency Range	Availability
440 - 510 MHz	Now
716 - 722 MHz (700 MHz Radio service)	Now
722 - 728 MHz (700 MHz Radio service)	Now
740 - 746 MHz (700 MHz Radio service)	Now
746 - 757 MHz (700 MHz Radio service)	Now
758 - 763 MHz (700 MHz Radio service)	Now
776 - 787 MHz (700 MHz Radio service)	Now
788 - 800 MHz (700 MHz Radio service)	Now
825 - 845 MHz (800 MHz Radio Service)	Now
870 - 890 MHz (800 MHz Radio Service)	Now
2300 - 2360 MHz (WCS service)	October 2012
3500 - 3700 MHz	October 2012
3600 - 4200 MHz	October 2012

Custom ISM/Amateur Bands

Frequency Range	Availability
420 - 450 MHz	Now
1240 - 1300 MHz	Now

Standard HauteSpot ISM/UNII Bands

Frequency Range	Model Option
902-928 MHz	-9
2192-2539 MHz 4800-6000 MHz	-M
2192-2539 MHz 2x2 MIMO 4800-6000 MHz 2x2 MIMO	-N
2192-2539 MHz 3x3 MIMO 4800-6000 MHz 3x3 MIMO	-NX
3650-3700 MHz	-37

All of above are available now